A FRAMEWORK FOR CROWD SIMULATION BASED ON THE JMONKEY GAME ENGINE

RAFAEL PAX

MÁSTER EN INGENIERÍA INFORMÁTICA. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin de Máster en Ingeniería Informática

Fecha
04-09-2017

Director/es y/o colaborador:

Jorge Gómez Sanz

Autorización de difusión

Rafael Pax

Fecha

El/la abajo firmante, matriculado/a en el Máster en Ingeniería Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "A framework for Crowd Simulation based on the JMonkey Game Engine", realizado durante el curso académico 2016-2017 bajo la dirección de Jorge Gómez Sanz en el Departamento de ISIA, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen en castellano

La simulación de multitudes juega un papel crucial cuando se trata del desarrollo de entornos inteligentes. La mayoría de los investigadores desarrollan simulaciones usando motores de juegos comerciales a través de los editores que éstos proporcionan. Esto dificulta el poder realizar una experimentación profunda sobre simulaciones de multitudes, y fuerza que la línea de investigación deba atenerse al paradigma de desarrollo propuesto por la herramienta. El objetivo principal del trabajo desarrollado es la contribución de un simulador de multitudes basado en 3D, con una arquitectura modular y extensible, adecuada para la experimentación con simulaciones de multitudes. Este framework se centrará de forma especial en la navegación y la coordinación de multitudes sobre un modelo realista del entorno, capaz de reproducir situaciones del mundo real. El simulador incluye implementaciones de algoritmos conocidos para el movimiento de multitudes, integrando también implementaciones de terceros.

El trabajo tiene en cuenta la necesidad de representaciones visualmente convincentes de la simulación más allá de las representaciones 2D, utilizadas regularmente en la literatura. Para ello, se contribuye con extensiones a herramientas de terceros que permiten importar texturas, animaciones y mallas que mejoran la calidad de la simulación.

El desempeño de la simulación se demuestra en un caso de estudio donde el desafío es encontrar una población cuyo comportamiento, dentro del simulador, reproduce un determinado tráfico entrante / saliente medido en áreas específicas de un edificio.

Este trabajo ha sido financiado por el proyecto MOSI-AGIL (S2013 / ICE-3019) a través de la Gobierno de la Comunidad de Madrid y Fondos Estructurales Europeos (FEDER).

Palabras clave

Framework de simulación, Simulación de multitudes, Grandes espacios, entornos inteligentes

Abstract

Crowd simulation plays a crucial role when it comes to the development of Smart Environments. Most researchers develop simulations using commercial game engines through the editors they provide. This prevents from a deep experimentation with the problems of crowd simulation and enforces to stick to the development paradigm of the tool. The main purpose of the work developed is to contribute with a 3D based crowd simulator framework that runs with a modular, extensible architecture suitable for experimentation with crowd simulation problems, such as steering or collision avoidance. This simulation framework will have an special focus on the navigation and crowd coordination within a model of a physical environment having the capacity of reproducing real world situations. The simulation framework includes custom implementations of known algorithms for steering and path smoothing and reuses implementations for collision avoidance.

The work takes into account the need of visually convincing representations of the simulation beyond the 2D regularly used in the literature. For this aim, it contributions with extensions to third party tools that allow to import textures, animations and meshes that improve the quality of the simulation.

The performance of the simulation is demonstrated in a case study where the challenge is to find a population whose behavior, within the simulator, reproduces a particular incoming/outgoing traffic measured in specific areas of a building.

This work has been funded by the MOSI-AGIL (S2013/ICE-3019) project through the Government of the Region of Madrid, and European Structural Funds (FEDER).

Keywords

Simulation Framework, Crowd simulation, Real time simulation, 3D engine, Large Facilities

Table of Contents

Agradecimientos

A todos los que, durante estos dos años me han enseñado que *lo bueno es enemigo de lo perfecto*, una y otra vez, me han puesto los pies sobre la tierra y han evitado que reinvente la rueda cada dos semanas.

En especial a Jorge, por su empuje, liderazgo y enseñanzas.

A Marlon por su increíble resistencia como *rubber duck debugger*.

A Juan, por otras tantas cosas que harían falta varias páginas.

Y por supuesto a Pilar, por su apoyo incondicional y aguantar mis monólogos sobre *bichitos, sistemas, componentes y mensajes* sin desfallecer.

Este trabajo ha sido financiado con fondos del proyecto MOSI-AGIL (S2013/ICE-3019), proyecto financiado por el gobierno de la Comunidad de Madrid y Fondos Europeos FEDER.

Chapter 1

Introduction

The concept of *Smart Environments* has evolved from its early beginning (late 1980s) [1] from primitive "computer walls" to environment designs equipped with sensors, actuators and computing components [2]. Although there are multiple definitions of what a smart environment is [2–4], they are commonly referred as context-aware design that adapts to the changes that occur in the environment in order to achieve certain objectives, such as optimizing energy consumption [5–8], activity recognition [9–12], or evacuation planning. Also, smart environments can be viewed as made of intelligent agents that perceive their environment through the use of sensors, and can act on the environment through the use of actuators, agents that constantly adapt their behavior to the behavior of the environment itself [13]. This system design could be centralized (all the data collected by the sensors is sent to a central management system), distributed (sensors and actuators are autonomous) or a combination of both.

Although there is some research about testing applications for smart environments [14–16], this is still a difficult task. It requires the installation of sensors and actuators, the communications and the software for the control system, and the participation of people who have to play the different scenarios. And the larger the environment, the more expensive the cost. More people is needed for performing the tests, and usually the number of devices required to test the installation increases too. This is costly, both in economic sense as well as in time. Also, there are some situations that cannot be tested for practical reasons,

such as evacuations, emergency situations or the restrictions imposed by the management of certain facility (e.g., airports, train stations). Repeating these tests is difficult and requires new investments, which makes the task of developing solutions for smart environments even more complicated.

The key elements here are the facility and people inhabiting that facility. If it was possible reproducing both within a simulation, the smart system could be attached to it and then, the developer, could start validating the smart system behavior. Reproducing the facility is not a major issue with current technologies, such as game engines, but reproducing the behavior of people within this facility is a still a challenging issue. Achieving this goal is the subject of crowd simulations.

1.1 Crowd Simulation

Crowd behavior has been studied for a long time. Crowd simulations using computational models are much more recent in comparison, and have attracted research from a broad range of disciplines, such safety science, robotics, physics, architecture or sociology. These simulations could be classified into two large groups: the realism of the behavior of the agents being part of the crowd (*behavioral realism*),an another one focusing on producing a high-quality visualization of the crowd (*visual realism*). In the first one, the behavior modeled tries to represent situations of the real world, and prove the validity of different types of sociological crowd models, crowd dynamics models or measure the evacuation time of a facility.

Simple polygonal representations of the environment, such as dots and squares drawn in a 2D window will suffice the visualization requirements of these simulations. In the second one, the main objective is having a believable, high-quality representation of the crowd. The interest and goals of this group is completely different (movies or video-games). What matters here is having a visual convincing result.

These two groups tend to converge as the technology advances. There is no need to choose

between visual or behavioral details because of a lack of computational resources thanks to GPU's [17–19], or distributed computing [20, 21]. Even if computational resources were available, achieving a high quality rendering combined with a believable and heterogeneous behavior of a large crowd is still challenging and an interesting research issue [22].

From the algorithmic perspective, there are still open issues, such as achieving a natural path navigation of the characters while steering, or reproducing the intuitive collision avoidance that humans show when walking within a crowd, as section 2 will show.

From the software engineering perspective, the construction of a crowd simulation platform is a demanding one. Beyond the basic simulation cycle, the structure and desirable features of a crowd simulator are not predetermined. The features a crowd simulation engine shows are [23]. This has led to different formulas to structure simulators, such as hierarchical structures [24] or [23].

1.2 Goals

The main goal of this work is to create an extensible 3D based crowd simulation framework that enables the simulation of crowds with a special focus on the navigation, crowd coordination within a model of a physical environment that can reproduce real world situations.

Crowd simulation is crucial to facilitate the development of smart environments, as explained in the introduction. The 3D capabilities are necessary to achieve the necessary realism that can convince external observers. The extensibility of the platform is a non-functional requirement that has been included because this kind of platforms cannot be the subject of just one piece or research. To create a line of research, the work has to be made in a way that new functionality can be easily incorporated. This constrains the way the architecture is conceived to follow a Entity Component System (ECS) [25], a data centric approach where systems process entities having certain associated components.

Note that a component here is not a *software component*. In this approach, a *component* labels entities existing in the simulation environment as possessing this particular aspect

(Depending on the language/implementations, could be Classes, Structs or Associative arrays).

The model of a physical environment represents the constraints and laws that the environment imposes. While 3D engines provide with rendering capabilities, physical interaction is realized by physical engines. A physical engine combined with tools for creating meshes that represents environment objects will enable to fast prototype and installation and to reproduce, to some extent, the real consequences of actions, such as collisions. Henceforth, this work will incorporate algorithms to deal with the already mentioned navigation/coordination problems.

To achieve this goal, the following activities have been proposed:

- Analyze the state of the art to review results in the addressed problems as well as determine the suitability of existing game engines.
- Identify requirements of services needed to address the crowd simulation.
- Design an architecture that realizes those services and that can be extended and that is built on top of the game engine.
- Implement solutions to the navigation, crowd coordination and activity execution within the architecture.
- Integrate tools for facilitate the definition of physical environments and characters.
- Evaluate the resulting system with a case study and several secondary developments.

1.3 Contributions

A first prototype was developed in the context of a degree project, MASSIS [26][27]. This prototype was developed using as a basis the Sweet Home 3D framework, a suitable approach to simulating 2D environments. Initially, it was considered as starting point, but the 2D representation was ineffective when dealing with obstacle avoidance in situations where the height was relevant or buildings with multiple floors were involved. Those are some of the

reasons why the effort was diverted towards 3D game engines and their use for building crowd simulators. These game engines provide convincing renderings of the simulations and allow additional functionality, such as generating different video streams of the same scenario corresponding to different points of view, or reproduce body gestures that are meaningful to the experts, such as nervousness or panic. Hence, the possibilities of 3D game engines to support crowd simulation are plentiful. The development of the framework has gone through three main phases, each one having as a result a different approach for simulating crowds. The first one, *MASSIS*, was the outcome of a degree proyect. The second one, *MASSIS2*, consisted on the first functional prototype of a crowd simulator using a 3D game engine. The current version is *MASSIS3*. It makes use of an extensible architecture, and it is intended to be the final version of this framework. Figure 1.1 shows this evolution.

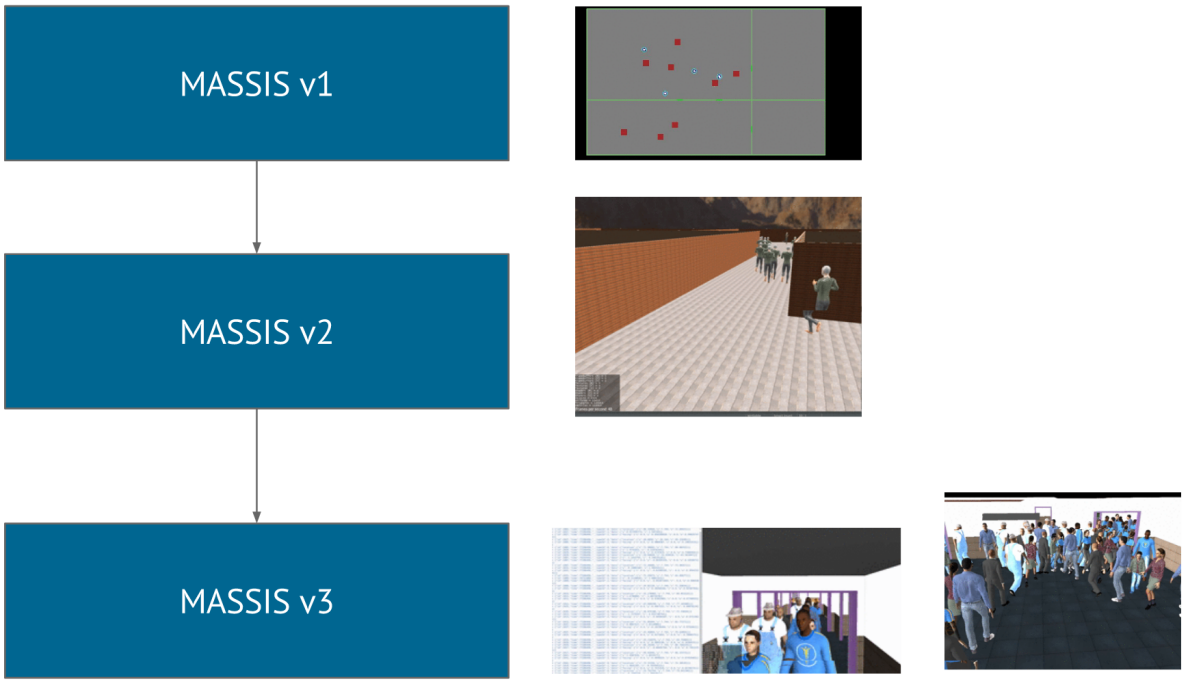

Figure 1.1: *Diagram showing the phases of the development of the MASSIS framework. The current version is MASSIS3.*

The work contributes with a simulation architecture built over JMonkey game engine, but it could be replicated over other game engines. For that aim, the work will include a

detailed explanation of each system addressing structural and behavioral aspects needed by the crowd simulation. The architecture strongly relies on the Entity Component System (ECS) pattern, a data driven alternative which is frequent in game engines because it allows to decouple state and functionality [25].

Multiple game engines were considered as starting point. From existing ones, JMonkey was chosen for this work because it was open source and was Java based. Using Java reduces the performance when compared to C++ game engines, but the development is less demanding and easier to debug. Also, the popularity of Java cannot be neglected. Different programming language rankings position Java always in the first places since a decade. Also, using Java gives access to a relevant number of resources.

Using this architecture, solutions for key problems in crowd simulation have been devised. In particular the work contributes to the following problems: navigation using paths and coordinating the movement of the crowd to reproduce evacuations and regular daily living activities.

In the area of visual realism, the work also contributes with some adaptations made to existing design tools to facilitate the creation of physical environments and import animations/gestures.

This work has been developed under the context of the MOSI-AGIL (S2013/ICE-3019) project funded by the Government of the Region of Madrid, and European Structural Funds (FEDER). The project is leaded by the Universidad Rey Juan Carlos, and participated byby Universidad Politécnica de Madrid and Universidad Complutense de Madrid. GRASIA research group represents the UCM in this consortium. The goal of the project is to create technologies that can influence pedestrians in their walking habits within a large facility.

The contributions of this work have been presented in several papers in international journals and conferences, which are appended as an annex of this document. They are summarized following:

- Rafael Pax and Juan Pavón. Agent architecture for crowd simulation in indoor envi-

ronments. *Journal of Ambient Intelligence and Humanized Computing*, 8(2):205–212, 2017 [27]. **JCR : 1.588, Q3**

- Jorge J Gomez-Sanz, Rafael Pax, Millán Arroyo, and Marlon Cardenas Bonett. Requirement engineering activities in smart environments for large facilities. *Computer Science & Information Systems*, 14(1), 2017 [28] **JCR 0.837, Q4**
- Rafael Pax and Juan Pavón. Agent-based simulation of crowds in indoor scenarios. In *Intelligent Distributed Computing IX*, pages 121–130. Springer, Cham, 2016 [26]
- Jorge J Gómez-Sanz, Marlon Cárdenas, Rafael Pax, and Pablo Campillo. Building prototypes through 3D simulations. In *Advances in Practical Applications of Scalable Multi-agent Systems. The PAAMS Collection: 14th International Conference, PAAMS 2016, Sevilla, Spain, June 1-3, 2016, Proceedings*, volume 9662, page 299. Springer, 2016 [29]. **This work won the first prize on the IBM award** of scientific excellence to the Best Demonstration presented during the 14th Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS'16).
- Rafael Pax and Jorge J Gómez-Sanz. A greedy algorithm for reproducing crowds. In *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection*, pages 287–296. Springer International Publishing, 2016 [30]
- Jorge J Gomez-Sanz, Rafael Pax, and Millán Arroyo. Towards the simulation of large environments. In *AmILP@ ECAI*, 2016 [31]
- Rafael Pax, Marlon Cárdenas Bonett, Jorge J Gómez-Sanz, and Juan Pavón. Virtual development of a presence sensor network using 3d simulations. In *International Conference on Smart Cities*, pages 154–163. Springer, Cham, 2017 [32]

1.4 Document structure

The document is organized as follows:

- In chapter 2 the state of the art is reviewed and discussed. The main tools for crowd

simulations are reviewed, and their different characteristics and issues are addressed.

- In chapter 3, the main components of the framework developed are explained, emphasizing the most important issues and how they are solved. The architecture emphasizes the role of the ECS pattern.
- In chapter 4, contributions graphic design tools that were used to increase the visual realism of the simulations are presented.
- In chapter 5, the crowd simulation tool is used to create simulations where the behavior of the crowd is parameterized off-line so that the resulting emergent behavior reproduces a particular pedestrian traffic.
- Finally, chapter 6 presents the conclusions and and indicates the next steps to be taken in the development of the framework.

Chapter 2

State of the Art

The state of the art is focused on the analysis of crowd simulation problems [22] addressed in this work. In particular, the problems considered are related to pathfinding, modeling influence of one character over others while moving (crowd movement), and the performance of activities while doing so.

Another subject of study is the selection of a game engine to support the crowd simulation. Many popular game engines exist today, among which Unreal, CryEngine, and Unity stand out. These, together with the chosen JMonkey are briefly introduced.

To complete the review, some existing crowd simulators are briefly checked to understand the value of this contribution.

2.1 Pathfinding

Pathfinding plays a key role in many real-world applications outside the scope of crowd simulation, such as video games [33, 34], robotics [35] or GIS [36, 37] and many more, or a conjunction of them [38, 39].

Although pathfinding methods are not a new issue[40], several methods and techniques have been developed and improved during the last decades, and the problem is still being a hot subject of research. However, the problem type has evolved, and the pathfinding algorithms take more factors into account. More recent works, such as [41, 42] takes into account the time and the positions of multiple agents.

Currently, the most common concerns are the combination of the algorithm's performance and realism: path generation techniques go far beyond classical, well-known pathfinding algorithms. Depending on the domain, the pathfinding might vary significantly: single-agent pathfinding, multi-agent pathfinding, that can be cooperative [43] or adversarial [44], with dynamic or static environments.

Normally, a path finding process is divided in two phases: The graph generation phase and the proper path-finding phase. Robotics and video games applications, such as the Recast toolkit [45], have been made an strong influence in the techniques for generating navigation graphs. Different kind of environments require different approaches for solving the problem. It is not the same a known environment as an unknown one, or a 2D, 2.5D (2D environment taking into account heights, e.g different levels) or pure 3D environments. The following subsections will try to explain briefly the most common pathfinding techniques, with their most representative advantages and their drawbacks.

2.1.1 Grids

A grid consists on a uniform division of the space into fixed size shapes. Usually, these shapes are squares, triangles, or hexagons, although the discretization of the space can be done in several ways in order to gain efficiency when executing pathfinding algorithms over them [46, 47]. These kind of representation is simple and easy to understand and implement, and it is widely used in 2D simulations [48–50], but has some memory and performance limitations that should be taken into account when using them [51, 52].

Representing the obstacles in a grid-based graph can be done in several ways, such as removing edges, removing nodes or assigning an infinite weight to the all the edges connecting a cell covered by an obstacle. Figure 2.1 illustrates how an environment could be discretized with a grid structure, marking cells as *obstacle* (darker ones) and *free* (light ones).

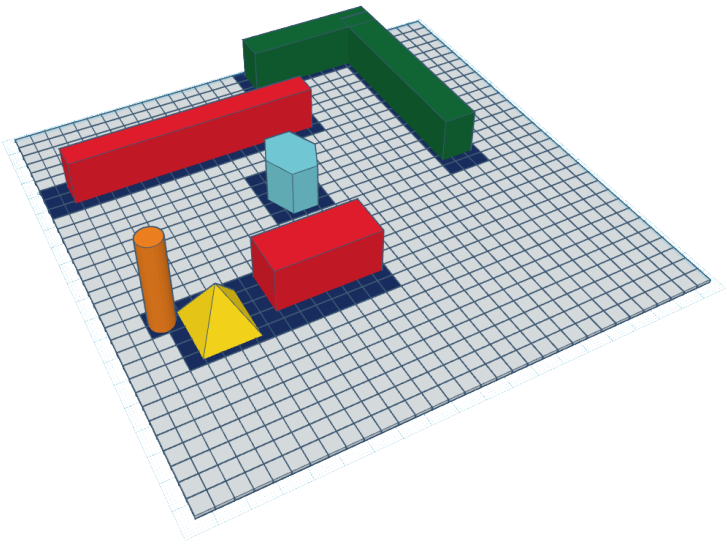

Figure 2.1: *Example of space discretization using grids*

2.1.2 Visibility graphs

A Visibility Graph [53] is a graph whose nodes correspond to geometric components, such as vertices or edges, and the nodes of this graph are connected only if there is not any obstacle intersecting the segment between those two points.

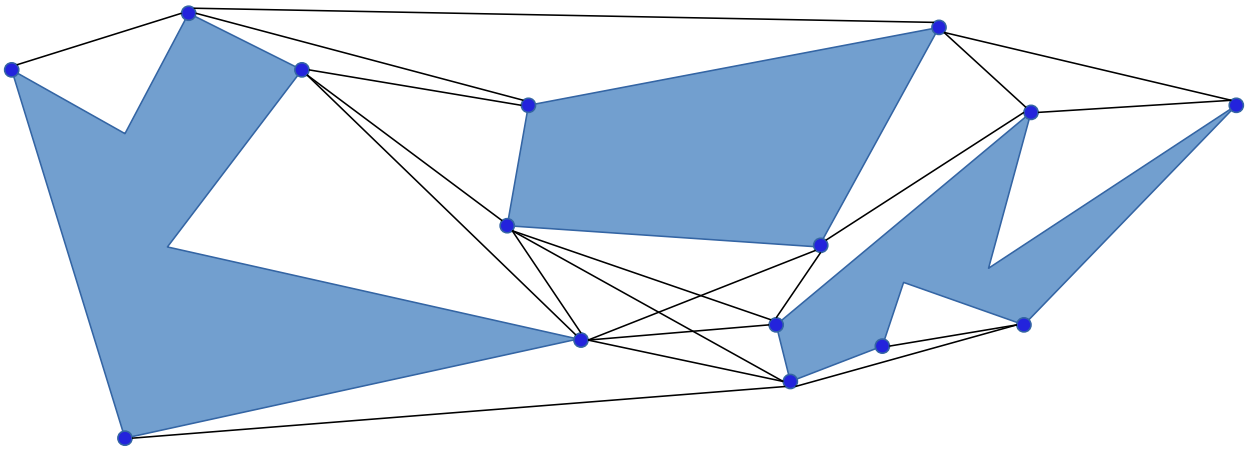

Figure 2.2: *Two dimensional visibility graph*

In a two dimensional plane, obstacles are represented as segments (usually forming convex polygonal shapes). Each edge of this graph represents a visible connection between

those two points (e.g They can see each other). Figure 2.2 shows an example of a visibility graph built over polygonal shapes, where every edge of the polygon is a node on the visibility graph.

Visibility graphs are a great choice for generating pathfinding graphs in 2D environments, but they become too complex in 3D environments. Usually, the 3D environment is simplified as a set of 2D environments in order to generate this kind of graph. Also, these kind of graphs can be easily integrated in a hierarchical pathfinding graph structure [26, 52] if the obstacle areas and walkable areas are known in advance, improving the patfinding process performance.

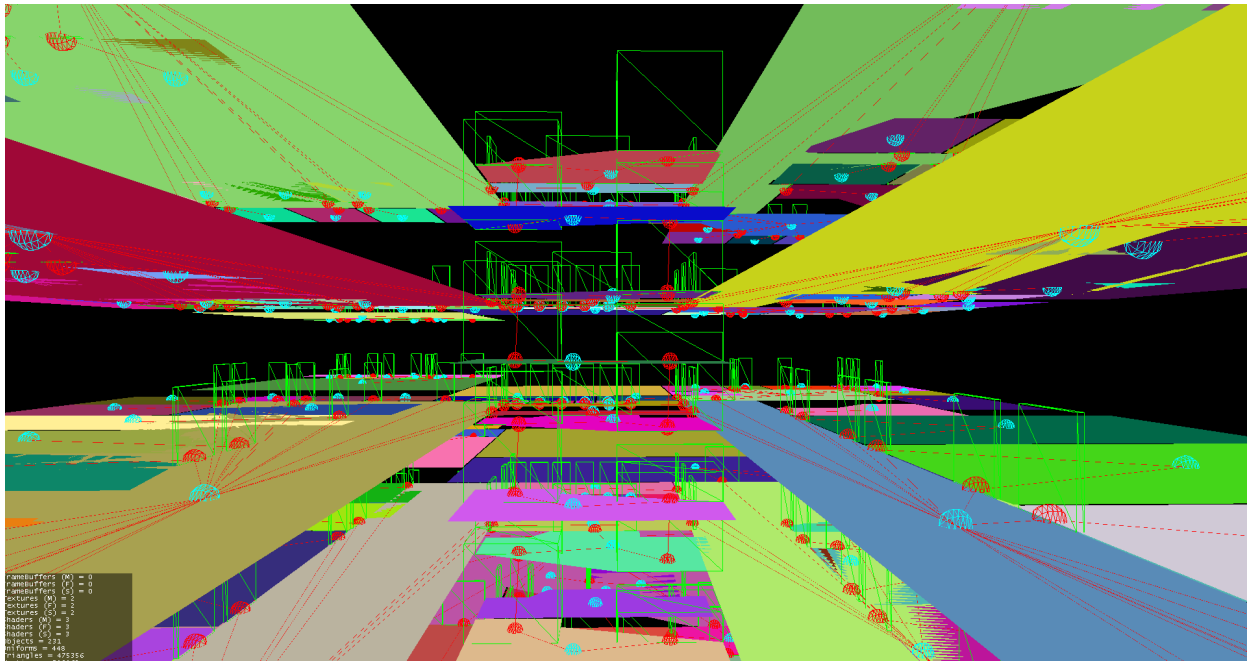

Figure 2.3: *Automatic room-door portal generation of the Faculty of Computer Science, UCM*

Although the approach proposed in [52] works well in 2D representations, it is not suitable in a three dimensional scene. When it comes to taking into account slopes and heights, too much adjustments should be made in order to make it work correctly. In the initial development of Massis3, this approach was taken and implemented (see Figure 2.3, but presented too much problems and was finally replaced with *Navigation Meshes*.

2.1.3 Navigation Meshes

A navigation mesh is a set of convex 2D or 3D polygons defining "walkable areas" of the environment (polygons must be convex in order to guarantee a free-walk of an avatar inside it). As opposite as visibility graph, which are based on the environment obstacles, navigation meshes are based on the environment free areas. With navigation meshes, the graph generation process becomes simple: the nodes of the navigation graph are the polygons, and the neighboring condition their adjacency. The quality of the mesh is conditioned by the polygon-generation algorithm used. Usually, the number of edges of the polygons generated by this methods vary between 3 and 6. In figure 2.4 a triangle-based navigation mesh is shown.

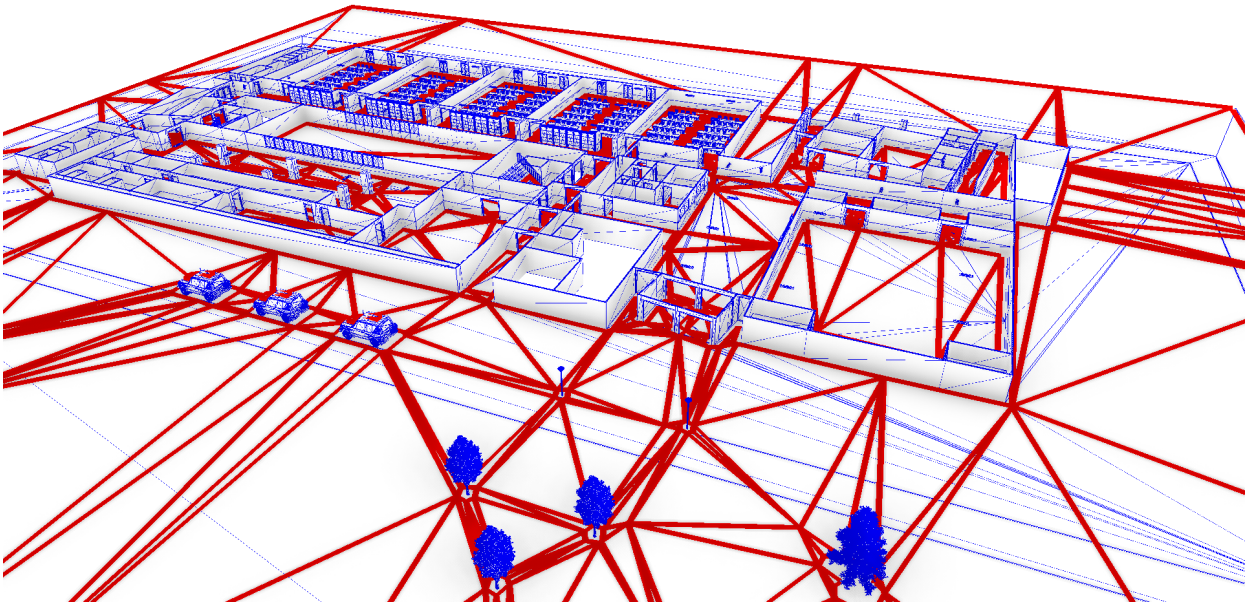

Figure 2.4: *Navmesh of the Faculty of Computer Science, UCM (first floor)*

2.2 Crowd Movement

Crowd movement models describes the emergent behavior based on the individual simulation of the participants in the crowd. Although the outcome of the activities performed by the crowd is heavily influenced by the path finding processes, dynamic obstacle avoidance processes should be taken into account, too. If the presence of other individuals is not taken into account on the path-planning processes, the resulting behavior might be unrealistic, specially for corridors or open areas. Classic path-finding techniques are useful for retrieving the *shortest path* from a point to its destination. If two individuals share a goal, and the path-planning strategy is to follow the shortest path, eventually they will take the same route, and will collide with each other. Hence, any realistic crowd model is the result of executing individual actions of the actors influenced by the presence or absence of other actors in the simulation. In this section, two of the most important approaches for modelling path-planning and obstacle avoidance will be explained: Force based and vehicle-obstacle based.

2.2.1 Force-based Models

Previous works have shown that modeling crowds using force models is quite effective. The work of Helbing et. al [54, 55] proposes a *social force* model, which considers the effect of each agent to all the others. The resulting social force affects the behavior of the agents, making them to change their direction. Although these kind of models are useful, they tend to be computationally heavy, as they compute the effect of the force that an element of the environment(an agent or an obstacle) on every other agent.

This approach can be a limitation for simulating large crowds. Modeling the forces with a force field, where the agents interact only with the field in their neighborhood, can alleviate such cost. The model proposed by Kirchner's [56] proposes an approach for modeling forces through a floor fields [57].

This model uses a discretized approach, where the environment is divided into a grid,

made of square cells, that can be occupied or empty. A cell can be occupied by exactly one pedestrian or by an obstacle. Each pedestrian can move to one of their four neighbor cells, in each time step. The individual can access to the floor field values around it, according to certain transition probabilities.

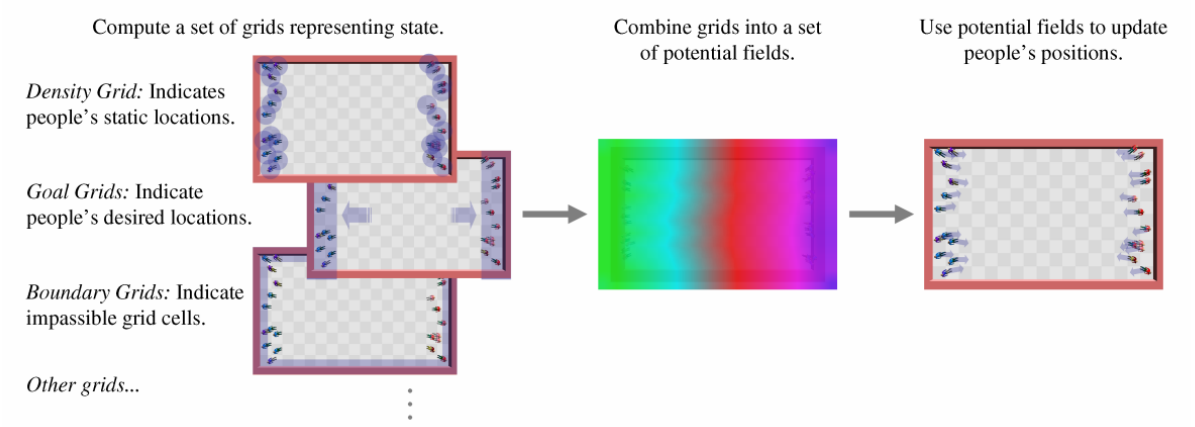

Figure 2.5: *Continuum crowds potential field navigation process*

Kirchner proposes the use of two floor fields, one static (S), and one dynamic (D). The static field do not change over time. It is used to specify areas in the environment that they are more attractive, such as an exit or an interesting point to the agent.The dynamic floor field is modified by the agents, which leave a trace in the field, and it is used for modeling the attractive interaction through agents. This field measures agent movement: the value of the cells of the dynamic field is increased while the agent moves through them. As time passes, the value of the cell is decreased, until reaches zero. Other crowd models follow the floor field paradigm, such as Continuum models (see figure 2.5) . They rely on a global potential field that conditions the movement of the individuals of the field. The potential field can be based on travel time [58] or in discomfort minimization [59]. Usually, these models are intended for outdoor environments or wide areas, but there are works showing how these navigation models can be applied to indoor environments [60].

2.2.2 Velocity-based Models

Force field-based models are not the only technique for simulating crowd movement. Non-grid based methods have been proved to be effective, too. Reynolds describe simple, but effective local collision avoidance methods [61–63], based on forward euler integrations. Also, velocity-based methods [64–66], have been used in many simulations and video games. These kind of models attract a lot of research interest, and variations of the same concepts are being developed actively, extending the velocity-obstacle concept to groups [67].

2.3 Execution of activities

The previous models mentioned before are usually used to simulate evacuation, or terrorism scenarios [68], where a crowd is aiming to exit a facility. If the behavior of the agent is more complex, the use of potential fields or velocity methods is not sufficient by themselves, or they add some restrictions to the behaviors that can be modeled [69] and another layer of abstraction should be used. Common, non-evacuation situations can be crowded, too. Train stations, supermarkets, hotels, university buildings, etc. are places where the individuals inside them have complex goals, and they need to perform different activities, but they are affected by the crowd, too.

Previous works have shown successfully how splitting the behavior in a hierarchy improves the quality of the simulation [70–72], in Massis3, we consider a behavior hierarchy based on three main layers:

2.3.1 Strategic behavior level

High-level reasoning models make easier the tasks for modeling an agent behavior, such as goal determination, activity scheduling or conflict resolution. Although these kind of behavioral modeling provides great expressiveness, is vary difficult to model lower-level behaviors, even the most simple ones, such as flocking. In this proposed behavior stack, the strategic intelligence chooses *what to do*, and delegates *how* to do it to one (or more)

crowd-level behavior models.

2.3.2 Crowd behavior level

Motion-level models provide a set lower-level behaviors for the individual, that can be switched by the high-level intelligence or even combined. Usually, while the building blocks of the strategic models are an abstraction of the environment, such as *The individual is seeing a fire*, crowd models take into account a more precise representation of the environment, with a geometry-level abstraction (distance, visibility...etc). Usually, this abstraction is made in 2D, or 2.5D (two dimensions taking into account height). These models run best from a global point of view, having all the information of the environment. However, if the presence of other individuals is not taken into account on the path-planning processes, the resulting behavior might be unrealistic, specially for corridors or open areas. Classic path-finding techniques are useful for retrieving the *shortest path* from a point to its destination. If two individuals share a goal, and the path-planning strategy is to follow the shortest path, eventually they will take the same route. In order to make virtual agents' paths become realistic, crowd density should be taken into account. Previous works, such as the analysis made by Mehdi Moussaïd et.al [73], based on the behavior of approximately 1500 pedestrian groups under natural condition, shows that the movement pattern changes depending on the crowd density of the area, due to the social interaction among individuals, so crowd density should be taken into account when modelling crowd behavior.

The emergence of bottlenecks is a common issue in crowded environments. However, when the bottleneck does not disappear in a certain amount of time, an individual may choose between different options, such as wait for the bottleneck to disappear, taking a different way for reaching its destination, or even pushing other individuals. This behavior could be modeled using potential fields, but may lead to unrealistic behaviors: going back and forth, oscillating movements, or falling into local minimums (although this can be avoided using internal agent states [74]). Also, velocity-obstacle models don't solve this

problem completely.

Waiting for the bottleneck to dissipate, or pushing to other individuals can be integrated in velocity models or floor field models. However, the decision of taking another way more similar as a human would do, does not fit easily in these models. Using a higher level of abstraction can ease the task of modeling the response of an individual to this kind of situation. This higher level of abstraction would take into account the crowd density repulsion force provided from the floor field, and would decide if a change of behavior is needed (e.g choose a route different route), or continue waiting/trying to pass through. The decision model does not specify how the agent has to move, but the crowd model that should follow in that moment.

2.3.3 Physical behavior level

The physical model is the bottom-most level in the behavior hierarchy. Implements and illustrates the environment elements, such as agent bodies, walls, doors, furniture and such. Also, this level is in charge of executing physical computations, such as gravity, or collision detection. Although the crowd model also has some kind of physical computations, like the application of forces to agents that make them not to bump each other, and obstacle avoidance methods.

2.4 Crowd Simulation Frameworks

There exist several crowd simulation tools, being many of them commercial or having a restrictive license, such as Legion [75], EXODUS [76], PedGo [76], STEPS [77] or Pathfinder [78]. But the vast majority of these simulation tools have a simplistic visual representation (with the exception of Pathfinder, as figure 2.6), or they are too much focused on concrete scenarios (such as evacuations). There are also open source simulators, such as PedSim [79] (two dimensional crowd simulator written in C) or CrowdMaster [80] (addon for blender written in python).

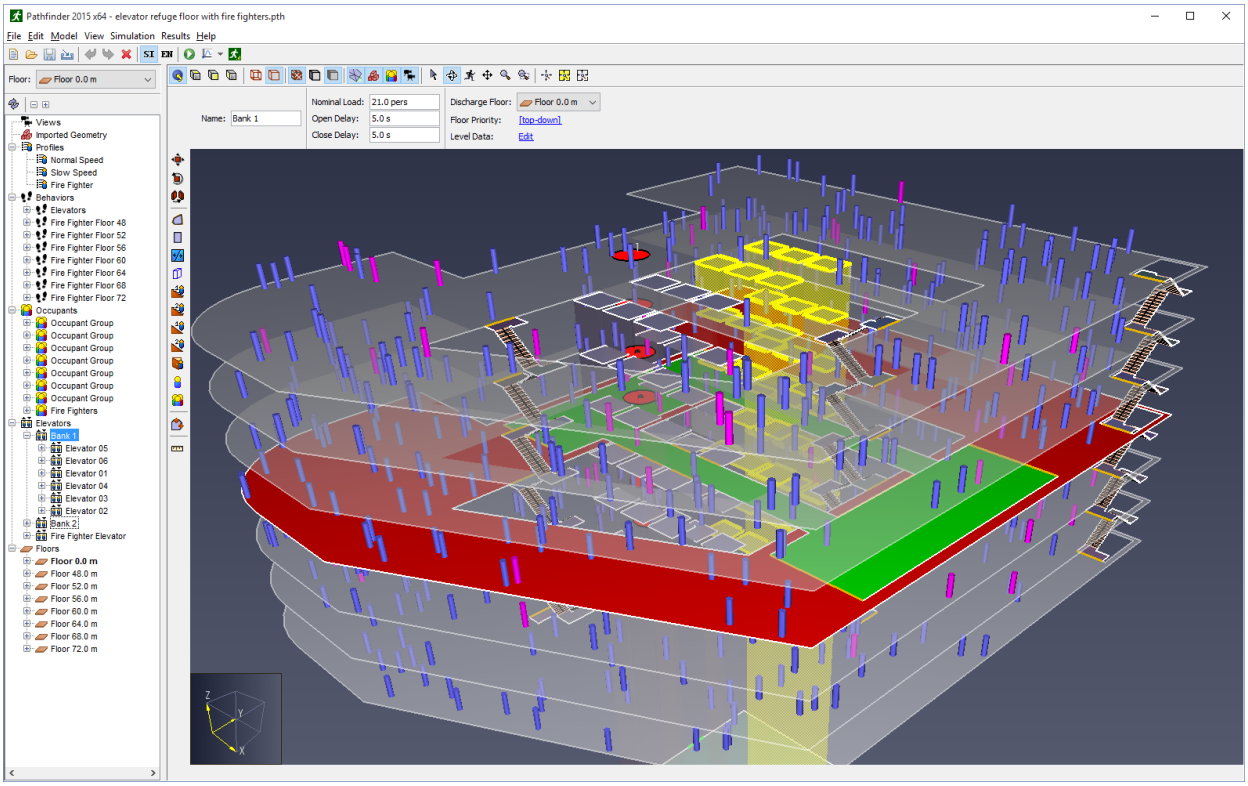

Figure 2.6: *Screenshot of the Pathfinder crowd simulator*

2.5 Popular Game Engines

The selection of a game engine for supporting crowd simulations is not an easy task. Nowadays, multiple game engines exist, being Unreal Engine, CryEngine, and Unity the most popular. Another one, not so well known is JMonkey Game Engine. These game engines will be explained briefly, and also the reasons of choosing JMonkey.

2.5.1 Unreal Engine

Unreal Engine (Epic Games) [81] provides a complete set of tools for developing high-quality 3D applications. Its current version, UE4, has advanced graphical capabilities, such as dynamic lightning, and a high-performance particle system. It has a good platform support. UE4 applications can be released for Windows, Mac, iOS, android, PlayStation 4 and Xbox One. It has a royalty of 5% whenever the application is released, being applied

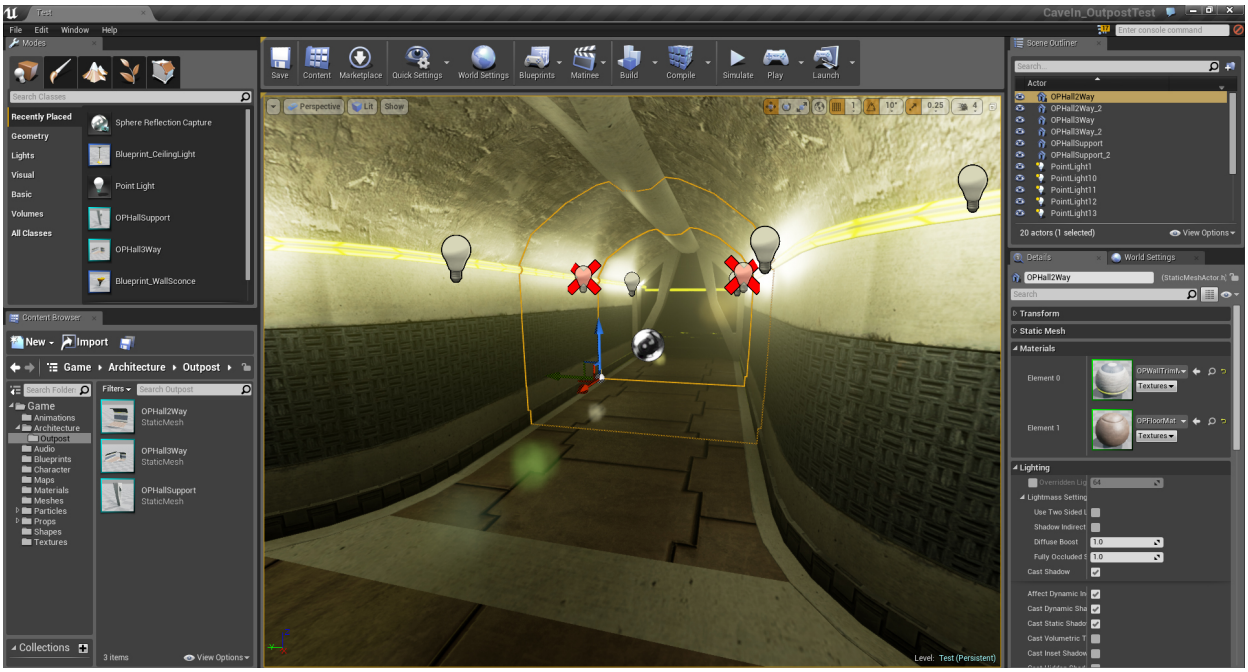

Figure 2.7: *Screenshot of the Unreal Engine 4 editor*

when the earnings of the game/application exceeds $3,000, making this kind of licensing a good choice for independent game developers, but it might need a negotiation for large companies. Although Unreal Engine provides access to its source code, this kind of licensing and the fact that its redistribution is not permitted makes it an non-open source project Figure 2.7 shows an screenshot of the Unreal Engine 4 editor.

2.5.2 CryEngine

The *CryEngine* (CryTek) was used for the first *Far Cry* game. Offering a multi-platform support (except for GNU-Linux), its graphical capabilities are on par with UE4. It has start-of-the-art lightning, a powerful and realistic physics, and an advanced animation system. Although it has a somewhat intuitive level-development GUI, the learning curve is still hard to hold if the developer has no previous experience with other game engines.

Its pricing model is different than other game engines: with a fee of $9.90 per month the application can be released. This can be a huge benefit for large studios. Figure 2.8 an

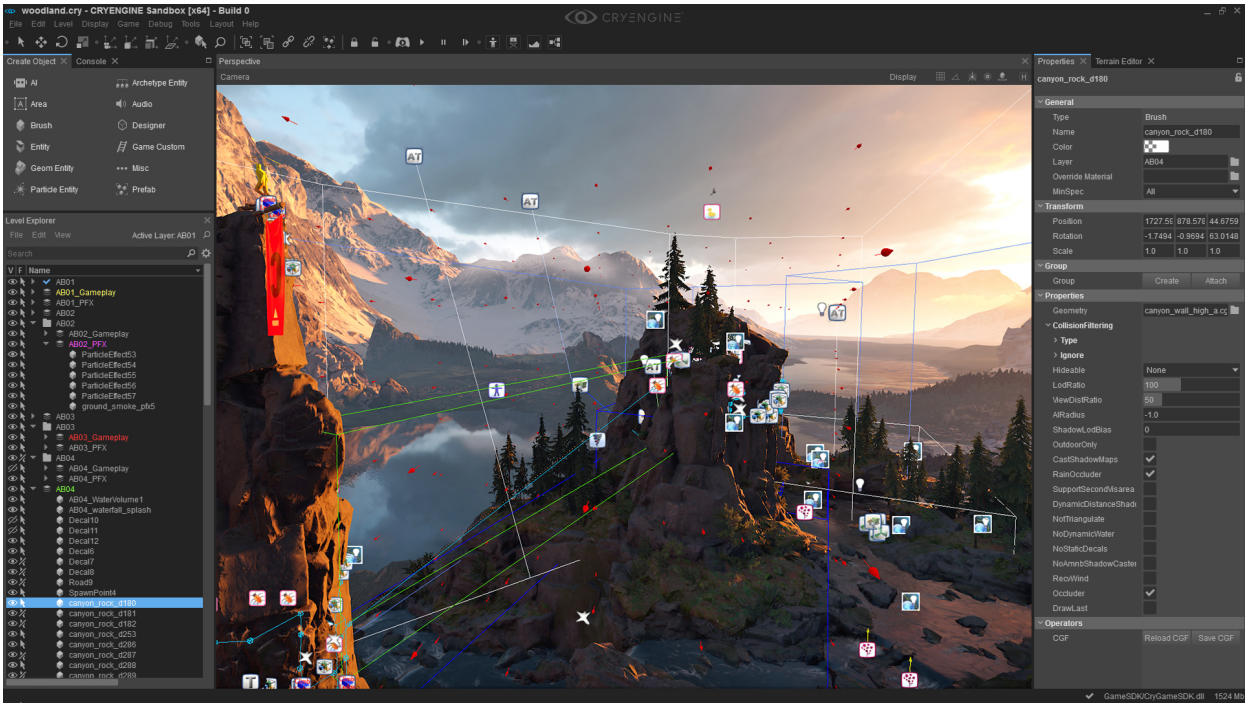

Figure 2.8: *Screenshot of the CryEngine editor*

screenshot of the CryEngine editor is shown.

2.5.3 Unity

The *Unity Game Engine* [82] (Unity Technologies) integrates a complete game-engine with a user-friendly SDK (see figure 2.9), making easy to develop 3D-based tools. The game logic is usually coded with C#, and provides access to common .NET libraries via Mono [83]. Also, provides physics capabilities, that can be run by the well known nvidia PhysX engine. The documentation provided by the Unity engine is extensive, provides a considerable amount of examples and it is backed up by community forums, making the learning curve smoother for beginners.

Another strong point of the Unity Engine is it's editor. The *drag and drop* editor beats by far the editors of other game engines. Unity projects are structured, and the scene graph editor allows to drag and drop scripts for defining the behavior of the elements of the scene and edit properties declared by those scripts in an easy manner. Also, the applications

Figure 2.9: *Screenshot of the Unity Engine editor*

developed with the Unity Engine can be run on Windows, OSX, Linux (still experimental), and as a Web Player. But Unity3D is propietary, and although it has started recently to release the source of some of its components, it is not open-source. It has two types of license: a limited free version, and a paid version. For essential development features such as version control, complex rendering or lightmapping it is necessary to pay. Also Unity3D leads to vendor lock-in: it is almost impossible to migrate a game written for Unity3D to another engine in case of requirement changes (e.g performance, licenses).

2.5.4 JMonkey Engine 3

JMonkey Engine[84] its an open source, multi-purpose game engine, which aims to help developers to create high quality video games using Java. The fact that JME is written in Java, is one of the most powerful advantages of using JMonkey Engine instead of other platforms is the ease of integration that this engine has with existing Java libraries. As the goal of this work is to create a crowd simulator, the requirements of the development differ

significantly from a 3D game, and the integration with third party dependencies is a must. Using Maven [85] or Gradle [86], this task becomes easier than ever. Mathematical, GUI, code generation or networking libraries can be integrated almost instantly. This fact speeds up the developing process significantly, and avoids the wheel reinvention.

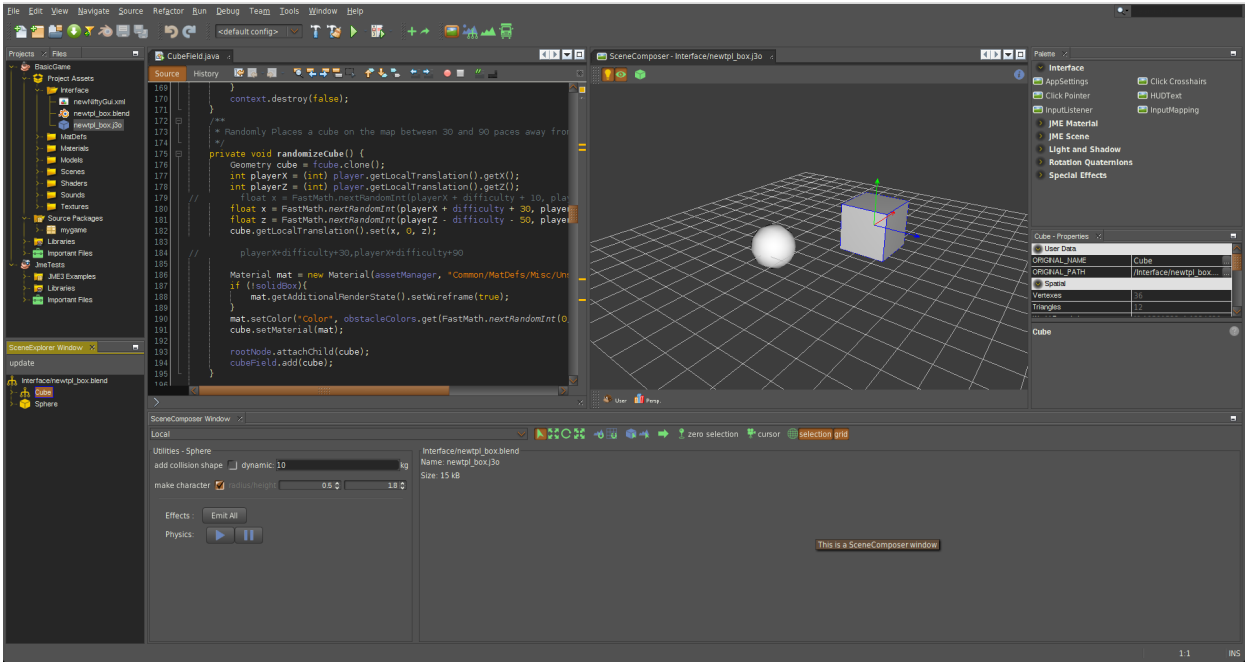

Figure 2.10: *JMonkey Engine SDK*

Although JMonkey Engine is presented as a *Game Engine*, it is easy to use it with different purposes. It can be used as such, but it is not as game oriented as other frameworks. The core package of JME3 provides some utilities that are very useful for developing games, but it is not a restrictive framework, and there is not a *correct way* for developing an application: The model proposed by JME3 is entirely optional and its architecture does not force developers to create applications following an specific pattern. Instead, it can be seen as a bunch of utilities packaged together that make the development of 3D applications easier. In fact, the proposed architecture of JMonkey it is not used in Massis3 (for example, the use of *Controls* or inheriting from *AbstractAppState*. Also, JME3 provides an SDK (Figure 2.10), although it is not as powerful as Unity's, speeds up basic modelling tasks.

2.6 Discussion

There are multiple options available when it comes to choose a tool (or a set of tools) for simulating crowds. Some of them are too specific to certain domain, a limited extensibility, or they come with a very restrictive license. As the framework is going to be fully open source, GPL3 licensed, many of them cannot be chosen. Finally, the JME3 engine was chosen, due to it's ease of integration with Java libraries and it's license (New BSD License).

Chapter 3

Framework design

The framework design for the crowd simulator is built over the JMonkey version 3 (JME3 from now on) architecture. The crowd simulator could just have just instantiated a typical JME3 game configuration to create the crowd simulator. However, there are some concerns that dis-encourage such path. These concerns have to do with the extensibility of the solution, the importance of working at the process level rather than the game character level, and the access to elements during the simulation loop. The impact of these requirements enforce some design decisions which are briefly introduced in section 3.1 and affect the core architecture as presented in section 3.2.

Those requirements enforce some measures, being the most relevant the adoption of the Entity Component System (ECS) approach [25] to structure the functionality of the simulator. This will lead to a set of systems that inter operate and provide the crowd simulation and visualization functionality. The resulting architecture design is presented in section 3.2.

3.1 Design Requirements

The simulation architecture proposed in this work has been designed to satisfy the following requirements:

1. There is a simulation/update loop that is continuously executed. Within this loop,

an update-render sequence is executed. Elements within the simulation can alter rendering related attributes (e.g. position, textures, meshes) but only when asked to do so. Rendering will assume the data is not modified while rendering.

2. The update loop should facilitate the joint operation over multiple entities in the simulation. This is specially useful when coordinating groups of characters in the simulation.

3. Elements in the simulation ought to be easily and efficiently obtained. During the update loop elements have to be obtained as fast as possible.

The game engine JME3 is designed to create individual elements in the simulation whose status is updated each cycle (requirement 1). This is suitable for a game design, but it turns out inefficient to research in crowd simulation. For that goal, elevating processes to first class citizens allows to see it the other way around: which processes to apply to the set of elements in the system to achieve the desired behavior each update cycle (requirement 2). This process based approach is demanding a fast access to the data required for each simulation element (requirement 3).

Therefore, a new architecture needs to be built over the JME3 framework. A first transformation is to account for process level constructs along the update loop (requirements 1 and 2). Addressing these two requirements withing an update loop cannot be done with just any pattern. Hierarchy based patterns, for instance, have problems as the number of behaviors to be considered. This problem is one of the arguments to enforce a composition based solution named Entity Component System (ECS) approach. The problems and limitations of the hierarchical approaches for update loops are introduced in section 3.1.1.

As a solution to this problem, and to satisfy above-mentioned requirements, a data driven approach for the design through the ECS pattern [25] is chosen. This decision is shared by many game engines [87]. This pattern permits the access to data in a simpler way at the same time that permits to interconnect elements. Following after this pattern, elements in

the contribution will be categorized either as Entities, Components, or Systems. The ECS pattern is introduced in section 3.1.2.

Applying this pattern is not trivial. A dictionary of entities will become inefficient with the wrong approach, because of the number of times the update loop is invoked. Components are added, removed and changed a lot of times during a single frame update and should be designed carefully for avoiding being a bottleneck in the simulation update loop. Standard JME3 access to elements is inefficient, so a different ECS based implementation is used in section 3.1.3. It is based on the Zay-ES [88] library which has been proposed as add-on to the JMonkey community by its developer.

The adoption of Zay-ES will enforce additional constraints over the update loop and new data structures created to accommodate the ECS approach. In particular, the incorporation of a specialized class whose purpose is to provide access to the defined entities. It should be remarked here that this will not create a hierarchy of entities in the Zay-ES approach. An entity will represent a collection of components, which will be the final data holders. Hence, an entity will need to be little more than an identifier.

Using Zay-ES has other advantages too, such as the possibility of collecting notifications of changes over stored entities and delivering them later on. Hence, a system can ask to the library to tell which changes were made since the last query.

This brings about another issue. If any system can access anything, extending the simulation in a safe way will require constant revision of which elements access to which others. Another secondary contribution of the work is the implementation of a visibility limitation mechanism for reintroducing some limitations to the access and management of elements, as introduced in section 3.1.4.

3.1.1 The update loop and hierarchy based approaches limitations

An update loop runs endlessly during the simulation/game execution. Each iteration, updates the simulation state, and optionally, renders some graphical output. Fairly simple, as

can be seen in Listing 3.1. Time control and more advances features have been omitted for clarity in this example. Usually, these features are provided by the game engine directly.

Let us assume a flock of birds is to be simulated. Its simulation loop would look something similar as the code shown in listing 3.2. The update of the simulation becomes tricky when considering the individual position of each bird in the flock. If the movement implies flying, it could be solved with an iteration over the elements. However, if multiple behaviors need to be accounted for each bird, e.g., eating or drinking, the solution becomes messy.

```
while(true)
{
    simulation.update();
    //optionally
    simulation.render();
}
```

Listing 3.1: *Simple simulation update loop*

The update loop could be rewritten in a more object-oriented approach manner, dividing the logic into *Processes* (also called *Managers*). A Process is nothing more than an interface exposing an initialization method, a method for notifying the iteration of the update loop and a tear-down method (see listing 3.3) The simulation loop logic can be broken, and moved into several managers.

Using processes, the code becomes more modular: The simulation loop calls the different `update()` methods of the processes, and each one executes their tasks (see listing 3.4). Then, each process can execute an specific task, such as rendering, computing physical forces, collisions, velocities and positions.

Depending on the application being implemented, the requirements of what is needed in an entity may vary. Informally, an entity can be defined as any object that exists in the game world.Usually, these entities can do several things, such as moving, following a path, changing its spatial representation due to an impact / action, particle emitting, perform an animation...etc. In a traditional OOP pattern, the way for representing these entities is to

```
void update()
{
    for(Boid boid: simulation.getBoids())
    {
        boid.calcNeighbors();
        boid.calcAlignment();
        boid.calcSeparation();

        boid.acceleration = boid.computeCohesion();
        boid.acceleration *= cohesionForce;
        boid.acceleration += boid.alignment *boid.alignmentForce;
        boid.acceleration -= boid.separation*boid.separationForce;
        boid.speed += boid.acceleration;
    }
    for(Boid boid: s.getBoids()) {
        boid.render();
    }
}
```

Listing 3.2: *Example of boids implementation in a simple simulation loop*

```
public interface Process
{
  void initialize();
  void update();
  void cleanup();
}
```

Listing 3.3: *Process interface example*

perform an object-oriented representation of the set of entities to be modeled.

The functionality required by the type of the entities must be encapsulated in an object, which can implement this functionality or delegate it to an inner attribute. As entities may have common functionalities (such as *move*), these functionalities are added as methods in a class being near the top-level hierarchy of an entity, being available to all derived classes. Although this is an advantage of OO approaches, quickly becomes difficult to maintain, an it is easy to end up with a *blob* object in some part of the hierarchy level.

The *blob* [89], or *God object* [90] is a common, well known anti-pattern, defined by a large single class (in terms of attributes or methods). Implementing functionalities in

```
void simulationInitialize() {
    for(Process process: simulation.processes)
        process.initialize();
}
void update() {
    for(Process process: simulation.processes)
        process.update();
}
void cleanup() {
    for(Process process: simulation.processes)
        process.update();
}
```

Listing 3.4: *Simulation loop using processes*

classes near the hierarchy root (in this case, the *Entity* class), results in an huge amount of functionalities on the leaf nodes, in most cases, unneeded. Also, if the implementation of these functionalities is done in the leaf nodes, another type of problem arises: Methods are only available to the leaf nodes, and this can lead to duplicated code, and forces the need of a refactoring.

3.1.2 Entity Component System

An Entity Component System (called also *ECS*) is a software architectural pattern which provides a way for splitting functionalities in terms of small and reusable parts. The main principle of ECS pattern is "composition over inheritance", which solves the problems caused by deep hierarchies, and offers the possibility of making performance improvements, due to its decoupled nature. Three elements are the key of an ECS [91]:

- **Entities:** Represent a concrete object in the world. Every discernible object should be represented by an entity. These entities lack of any type of data and methods. They are similar to objects in OOP, in the way that if in the world there are 10 cubes, should be 10 entities. Informally, an entity is nothing more than some kind of tag which expresses the existence of something that can be treated individually. Entities alone are somewhat useless: they should contain *Components*.

- **Components:** Every entity has different aspects that define some properties and specify in which way the entity interacts with the environment. These aspects will define the behavior of the entity, that will be managed by *Systems*.
- **Systems:** The systems are very similar to the *Processes* mentioned before: They are in charge of executing the different parts of the logic of the simulation. Systems encapsulate the necessary logic to perform a change in the simulation. Systems are subscribed to the changes performed on the entities, relying on their components to execute their tasks.

With these elements, the update cycle can be revisited as the pseudo-code from listing 3.5 shows. The idea is that each system will retrieve those entities that have the necessary components to perform the function of the system. For instance, if the system is in charge of moving all humans, the system will retrieve those entities that have a human component and perform the necessary transformation.

```
for each system or process to be applied,
    do ask the system to perform the update
        (within each system)
        for each entity that has the necessary components, do
            apply the process to the entity
```

Listing 3.5: *Revisiting the update loop*

3.1.3 Basic changes to JME3 to conform ECS design approach

In principle, it is possible to use a vanilla version JMonkey using ECS principles, but it is not recommended because of the inefficient access of JME3 to elements.

To realize the ECS approach, each update cycle, the JME3 would have to retrieve entities having a certain associated component type. JME3 solves this with a list that contains all the components (in JME3 are called *Controls*) of an Entity (in the case of JME3 is an *Spatial*), and every time a component is added, retrieved or removed the list must be iterated. The computational cost of these operation is then $\mathbf{O}(n)$. Listing 3.6 shows the code for this

approach in JME3. In Massis3 entities are modified multiple times in each update loop iteration, so a complexity of $\mathbf{O}(n)$ is not acceptable.

```
public <T extends Control> T getControl(Class<T> controlType) {
    for (Control c: controls.getArray()) {
        if (controlType.isAssignableFrom(c.getClass())) {
            return (T) c;
        }
    }
    return null;
}
```

Listing 3.6: *Retrieving a control in JME3*

To improve this, an ECS specific implementation is used, the Zay-ES library [88], but extensions have been developed in order to improve how components are managed, and to avoid bugs. This extension is a secondary contribution and it is introduced in section 3.1.4.

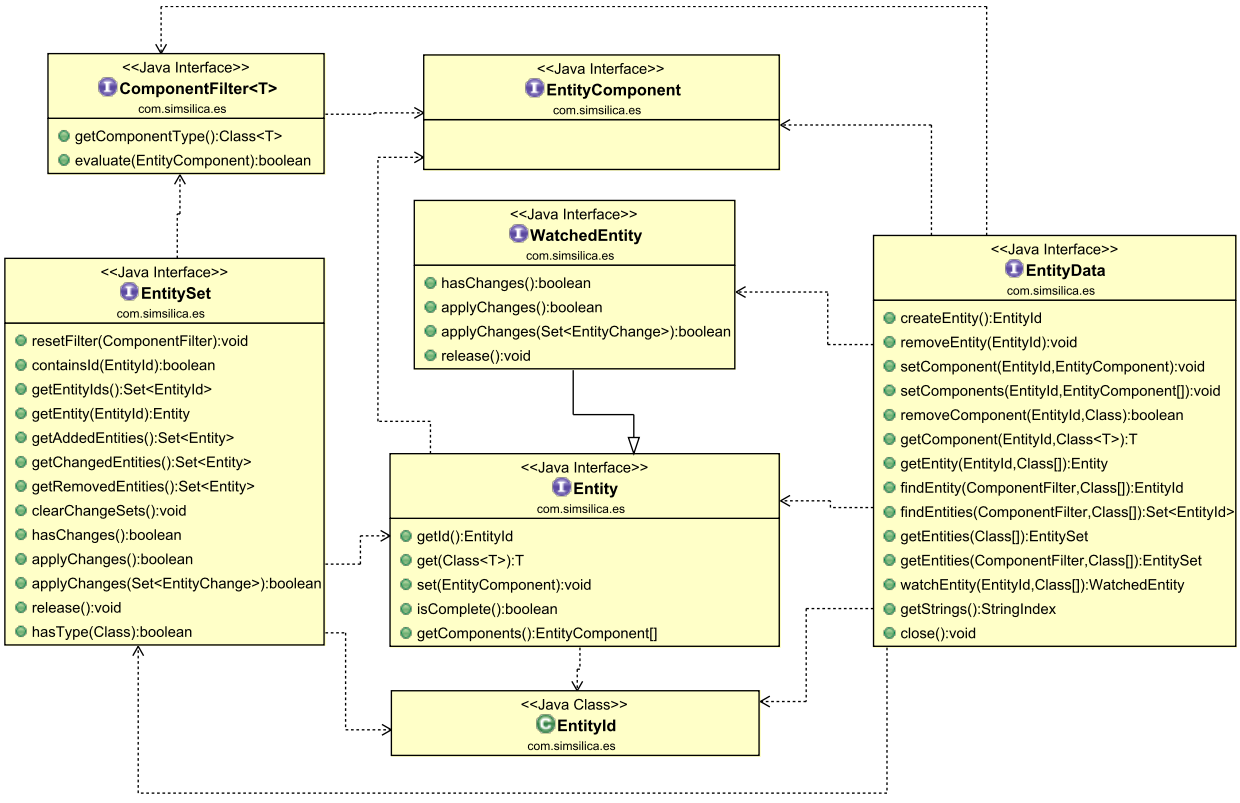

Figure 3.1: *Zay-ES Entity data model*

`EntityData` acts as an entry point. An `EntityData` instance can provide references to

any entity and perform different kind of searches, like obtaining the set of entities that have a certain component type. A developer will have to determine before hand how many entities are going to be used. It is not expected to define a hierarchy of entities, just invocations to `EntityData` to obtain an identifier to which attach components afterwards.

The `EntitySet` is used to aggregate entities in these situations. There is an `Entity` interface, but a development will not produce a hierarchy of implementations of `Entity`. A developer will ask the `EntityData` to create an entity and the developer will determine its use as a particular simulation element. Changes over the entity can be observed and a handler, `WatchedEntity`, will be obtained. The data associated to an entity will be divided into different components. Each component will hold data used with a specific aim. For instance, a position component will hold the 3D position of the element in the environment.

The default approach taken by the Zay-ES library for storing entities and their components (`EntityData`) is using one component handler per component type (usually an implementation that is backed up by some kind of `Map`, with constant time access). Doing it this way, the access to the components requires almost constant time. Listing 3.7 illustrates how this is achieved, and the most important elements of the architecture are shown in figure 3.1.

```
@Override
    public <T extends EntityComponent> T getComponent(EntityId entityId, Class<T>
        type ) {
        if(entityId == null ) {
            throw new IllegalArgumentException("EntityId cannot be null.");
        }
        ComponentHandler handler = getHandler(type);
        return (T)handler.getComponent(entityId);
    }
```

Listing 3.7: *Accessing a component in Zay-ES*

For tracking entities matching a particular aspect (e.g having Position and Facing), *Entity Sets* are used. Every time component is changed, the entity set gets notified about that change, using a `WatchedEntity` instance that can be queried later (usually by Systems).

It can be seen as the *Observable* pattern, but the update must be checked manually. This is useful for controlling the update flow of the simulation, or for checking changes with different time intervals. The most common pattern for dealing with EntitySets inside an update loop is shown in the Listing 3.8. A System can have multiple entity sets, each one matching different aspects of an entity.

```
public void update() {
    if (entitySet.applyChanges())
    {
        entitySet.getAddedEntities().forEach(e->/*...*/);
        entitySet.getChangedEntities().forEach(e->/*...*/);
        entitySet.getRemovedEntities().forEach(e->/*...*/);
    }
}
```

Listing 3.8: *Usual way for dealing with EntitySets*

It can be inferred that a problem of this approach is that `EntityData` gives arbitrary access to any registered entity. Anyone can retrieve an entity, obtain the corresponding component and change the stored data. It would be even possible to remove alter associated components to an entity. This problem will be considered in section 3.1.4.

3.1.4 Controlling the access to entities

`EntityData` acts as an entry point for the components of any entity. To add a control layer that determines who access what, two interfaces have been added `Entity Component Modifier` and `Entity Compoment Accessor` to act as mediators to any access operation made to an entiy. The mediators are introduced through a specific `EntityData` interface implementation, called `EntityDataSystem`, are shown in figure 3.2.

When some element requires access to an entity, it will be made through a `Entity Component Accessors` implementation as figure 3.3 shows. The implementation will determine if the operation should continue or not. This additional check generates an undesirable overhead. Its purpose is to enforce good practices, and can be disabled for production environments.

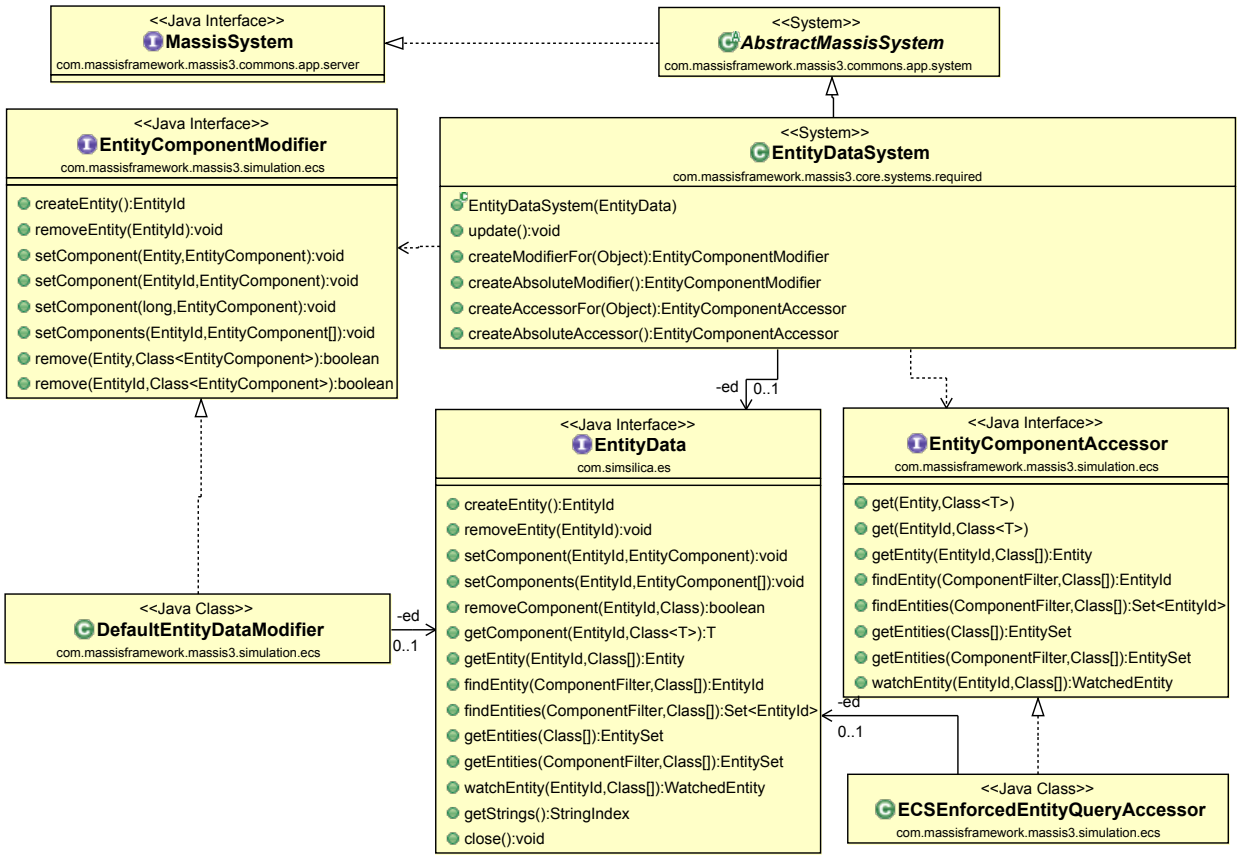

Figure 3.2: *Entity Data System class diagram*

This accessor scheme has another important use. As the codebase of systems grows, tracking the components that each system manages becomes complicated. Maybe two systems are changing `Position` components in their update loop. Or maybe, due to a typographical error, a System is inserting/modifying/removing an erroneous component(e.g rotation instead of position). This work as addressed this problem creating annotations declaring if a class is allowed to access, remove or modify certain types of components. Listing 3.9 is an example of this kind of declarations for the RVO2System which will be reviewed in section 3.2.

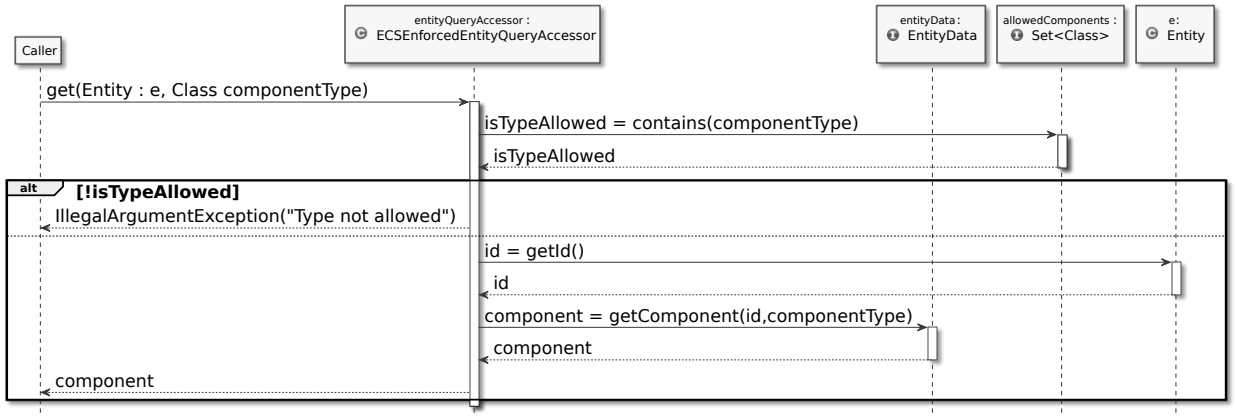

Figure 3.3: *Sequence for obtaining a component using an EntityComponentAccessor*

```
@TracksComponents({
        RVO2Component.class,
        Position.class,
        Speed.class,
})

@GeneratesComponents({
        RVO2DesiredDirection.class
})
```

Listing 3.9: *Example of annotations declaring what kind of components is managing a system*

3.2 Architecture

Taking into account the requirements from section 3.1, a core architecture is proposed in figure 3.5. This figure 3.5 shows elements from JME3 in white, while contributed elements necessary for the crowd simulator appear in yellow. To facilitate the recognition of ECS pattern elements, class stereotypes have been used to denote elements playing the role of System and Component. There will be no elements represented as entities explicitly because there no hierarchy of entities, as explained in section 3.1.2. Hence, there is only one entity type. Nevertheless, it will be explained in each case which logical entities are needed.

The simulator will be represented by a *ServerJMEApplicationImpl* and extends a basic JME3 application, which basically initiates the game engine with the game parameters.

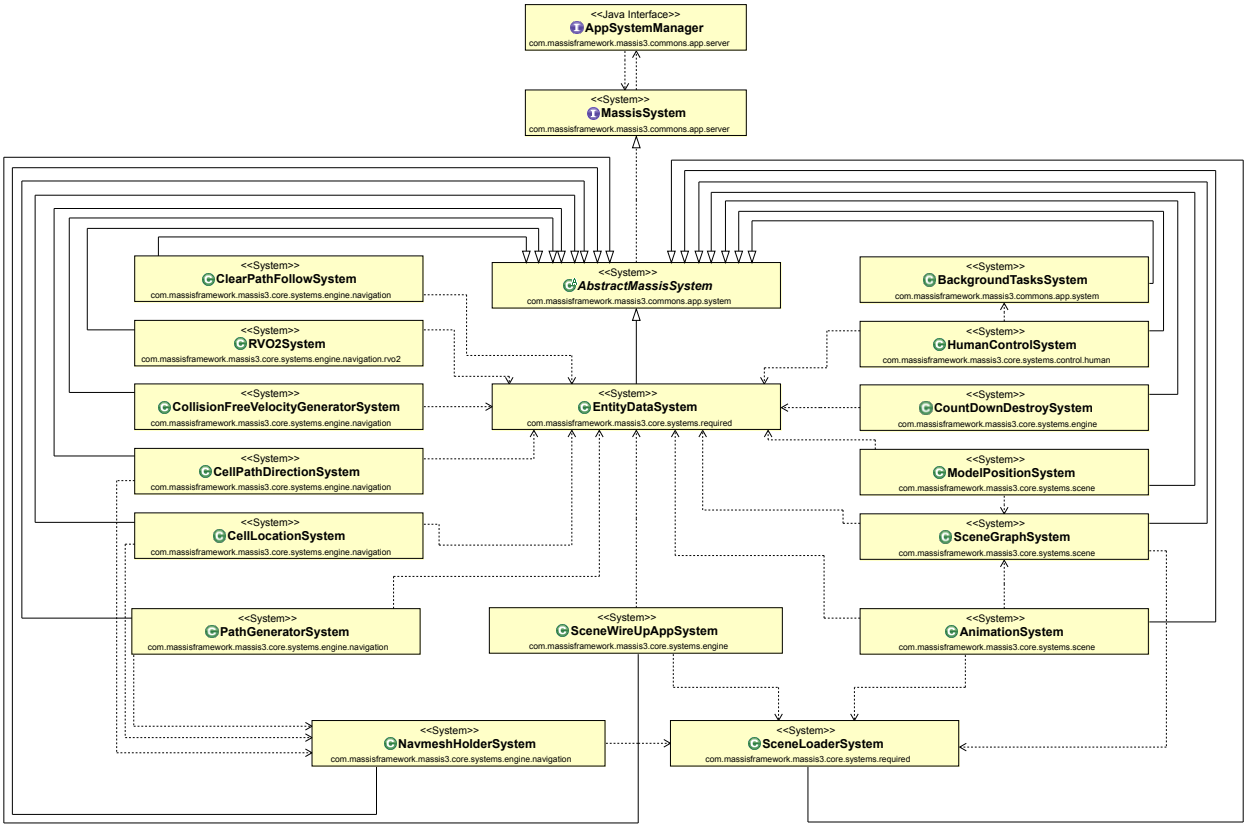

Figure 3.4: *Overview of the framework systems and their relationships*

The initialization consists on defining logical entities, the associated components, and the systems that will be processed. The modified update loop according to requirements 2 and 3 is allocated within *ServerJMEApplicationImpl* too.

The systems, as accounted by the ECS approach, are managed by the *AppSystemManager* interface. here is a basic implementation, *AppSystemManagerImpl* that contains the available systems which can either implement *MassisSystem* or extend *AbstractMassisSystem*. Each system will represent a process executed over the set of entities that have a particular associated component type.

Through the *AppSystemManager*, each update cycle all instantiated systems have to be invoked. Basic systems are introduced in figure 3.4. All systems extend the AbstractMassisSystem and have access to the EntityDataSystem. The later provides access to the entities defined in the simulator, as required by the ECS pattern. The purpose of each system is

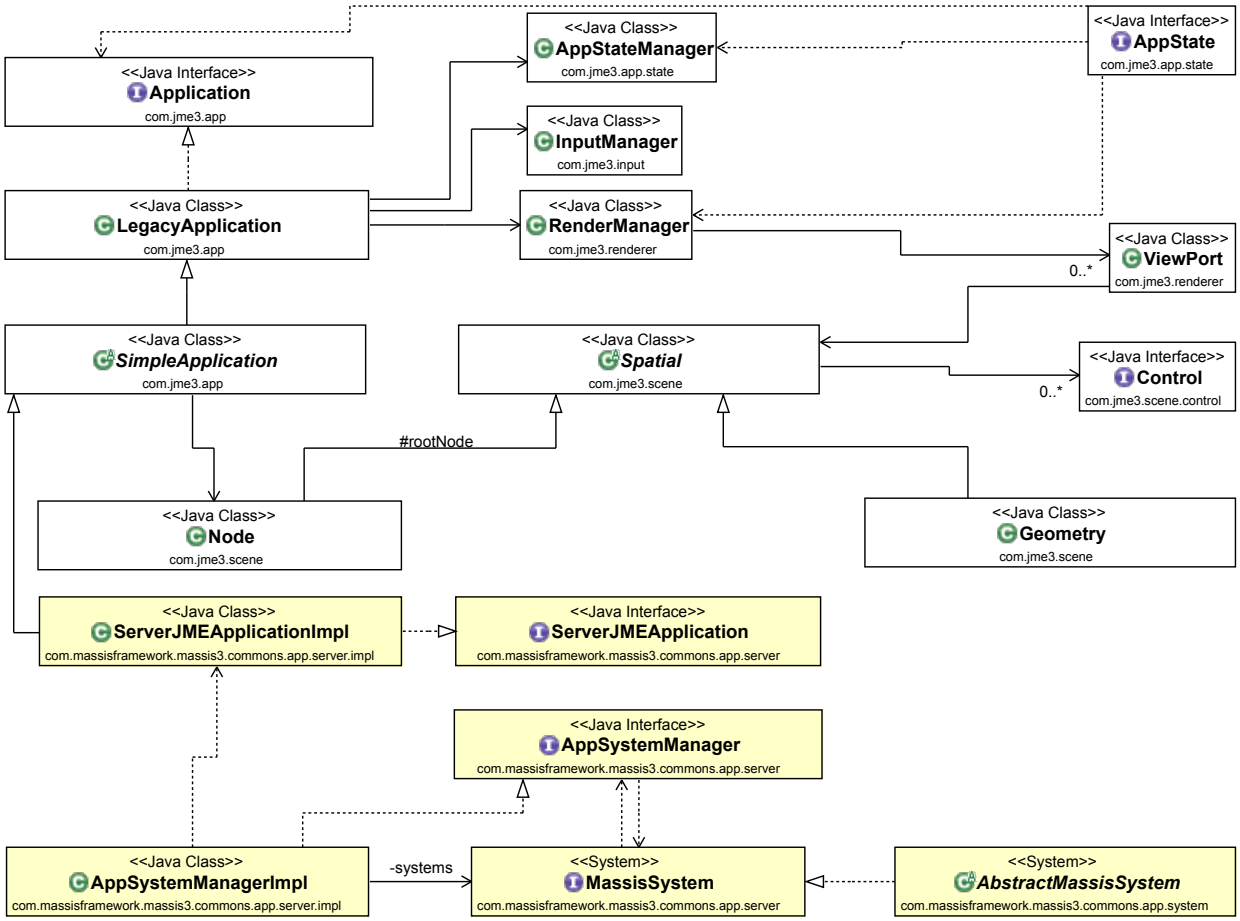

Figure 3.5: *Relationship between JME3 and the framework developed*

listed following:

- ClearPathFollowSystem: Moves virtual avatars through a collision-free path.
- RVO2System: applies the RVO2 algorithm for dynamic obstacle avoidance.
- CollisionFreeVelocityGeneratorSystem: computes collision-free velocity vectors.
- CellPathDirectionSystem: Given a path for an entity of the crowd, generates a direction to be followed, based on the navigation mesh.
- CellLocationSystem: Locates points in the navigation mesh.
- PathGeneratorSystem: Generates paths for entities.

- NavMeshHolderSystem: Holds the navigation mesh.
- SceneWireUpAppSystem: Is in charge of setting up the simulation environment given certain configuration. Performs scene loading tasks, and creates the entities of the simulation.
- SceneLoaderSystem: Entry point for loading assets for other systems.
- BackgroundTasksSystem: Maintains an internal thread pool for executing tasks outside the simulation update loop.
- HumanControlSystem: Adds a layer of abstraction for managing human entities (creation, deletion, movement, animation), hiding ECS complexity. It is intended for be used by extensions.
- CountDownDestroySystem: Tracks and destroys any entity having a `CountDownDestroyComponent` (which specifies a timeout).
- ModelPositionSystem: Manages the position of the entities in the JME3 Scene graph.
- SceneGraphSystem: Mantains the consistency between entities and their 3D representation in the JME3 Scene graph.
- AnimationSystem: Performs animation tasks.

The most relevant are those in charge of the scene (Scene Graph System, the animation system, the model position system), the scene loader, those in charge of the navigation (cell location system, Path generator system, RVO2 System, Steering Direction System, Collision free Velocity Generator System, and Clear path follow system). These will be detailed following to provide more details on their functionality.

3.2.1 Graphical Systems

These systems are in charge of the representation of elements in the scene. They have a direct communication with the JME3 Engine, and they act as a *bridge* with it. Other systems should access these systems (mainly the SceneGraphSystem, explained in subsubsection 3.2.1) for dealing with the JME3 representation of the entities. (`Spatial`, `Material`,`Animation`, etc.).

Scene Graph System

This system tracks entities **having material (MaterialInfo) and 3D model description (Model3DInfo) components**, and manages the representation of them in the JME3 Engine through the `Spatial` element.

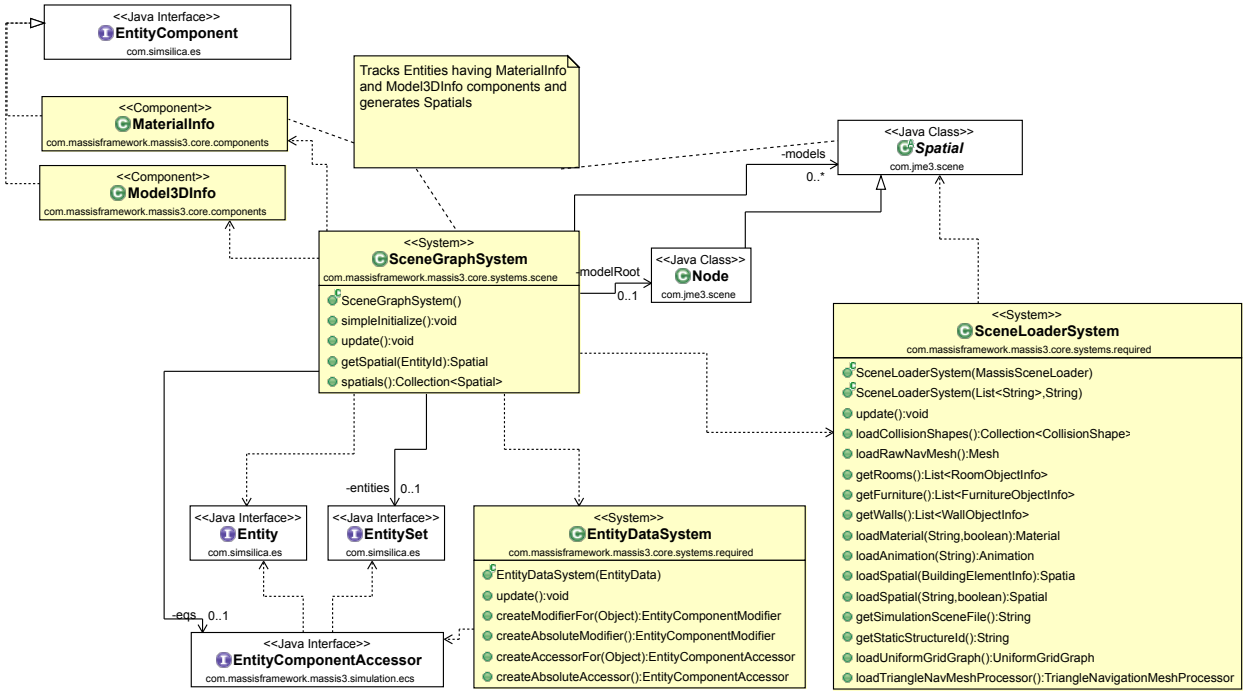

Figure 3.6: *Class diagram showing the relevant relationships of the Scene Graph System*

It is in charge of maintaining the consistency between the entities having a 3D representation and the JME3 Engine. In a way, it transfers information stored in the components

to the JME3 data structure used for rendering, *Node* and *Spatial* classes. Figures 3.6 and 3.7 show the class diagram and the sequence diagram of this system, respectively.

According to figure 3.6, the presence of `MaterialInfo` and `Model3DInfo` means that should be rendered in 3D. This systems loads its corresponding assets and maintains the Spatial/Material-Entity correlation in a Map. When the entity gets deleted (or it's *Model3DInfo* component), this system removes it from the JME scene graph.

The sequence of steps is presented in the diagram from figure 3.7. Initially, the EntitySet informs which changes have occurred (added, changed or removed entities). For removed entities, the corresponding JME3 elements are removed too. For added elements, their corresponding *Spatial* (the one specified in the `Model3DInfo` component) is loaded by another system (SceneLoaderSystem) and then is attached to the system root node, so that the new entity can start being rendered.

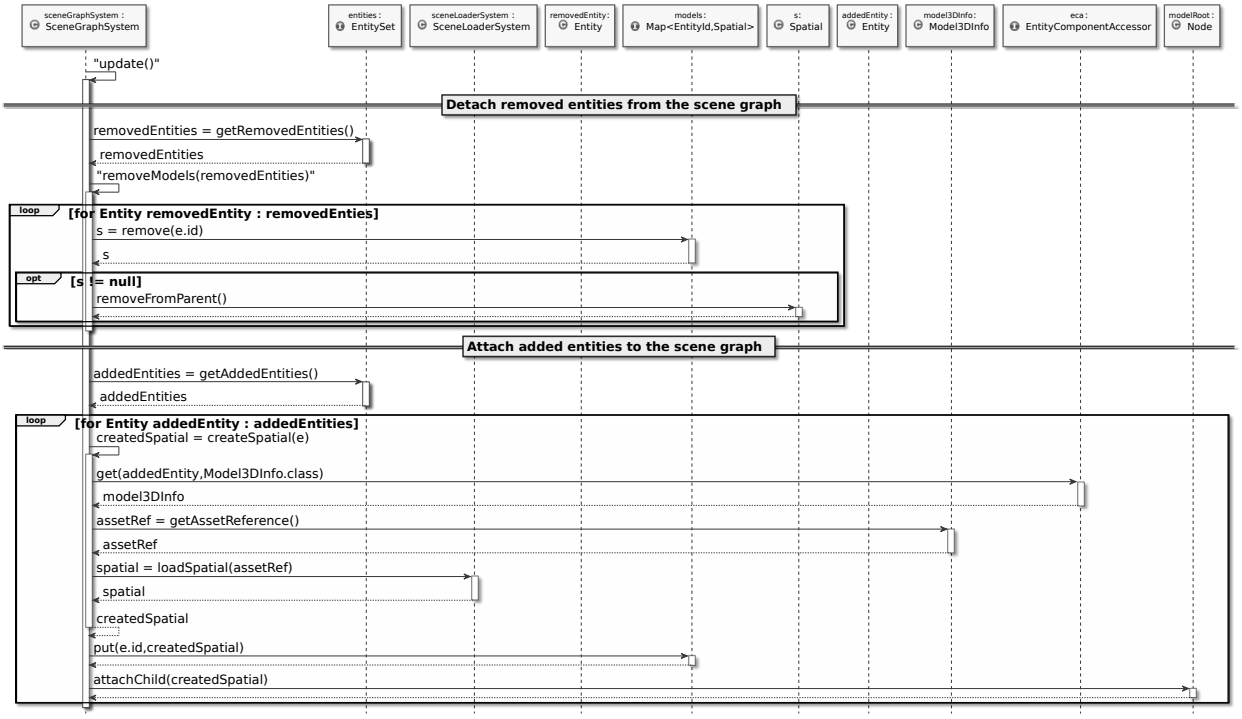

Figure 3.7: *Sequence diagram of the addition of 3D models to the scene graph done by the Scene Graph System*

Model Position System

The *Model Position System* is in charge of synchronizing the logical transform of the entities. The entity's trasform is represented as a combination of a `Position`, `Scale` and `Facing` component) and their associated spatial in the JME3 scene graph. This system uses `SceneGraphSystem` for retrieving the associated entity's spatial, and sets the correct transform on it. Every time the position, rotation, or scale of an entity having a 3D model(`Model3DInfo`) associated is changed, the transform of their corresponding `Spatial` is updated. This process is illustrated in figure 3.9

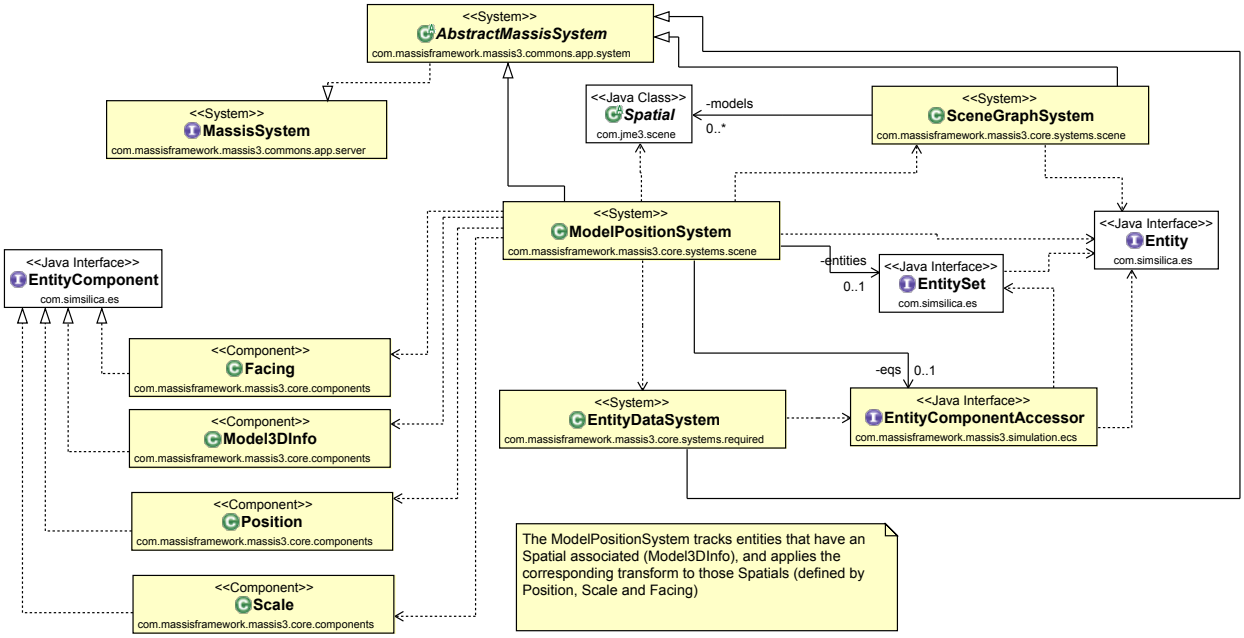

Figure 3.8: *Model Position System class diagram*

Initially, all the scene graph responsabilities were located in `SceneGraphSystem` in order to have all the JME logic code in one place, but other Systems, such as debug systems, required direct acess to the entities spatials, so the original `SceneGraphSystem` was divided into three systems (Scene Graph, Model Position and Animation).

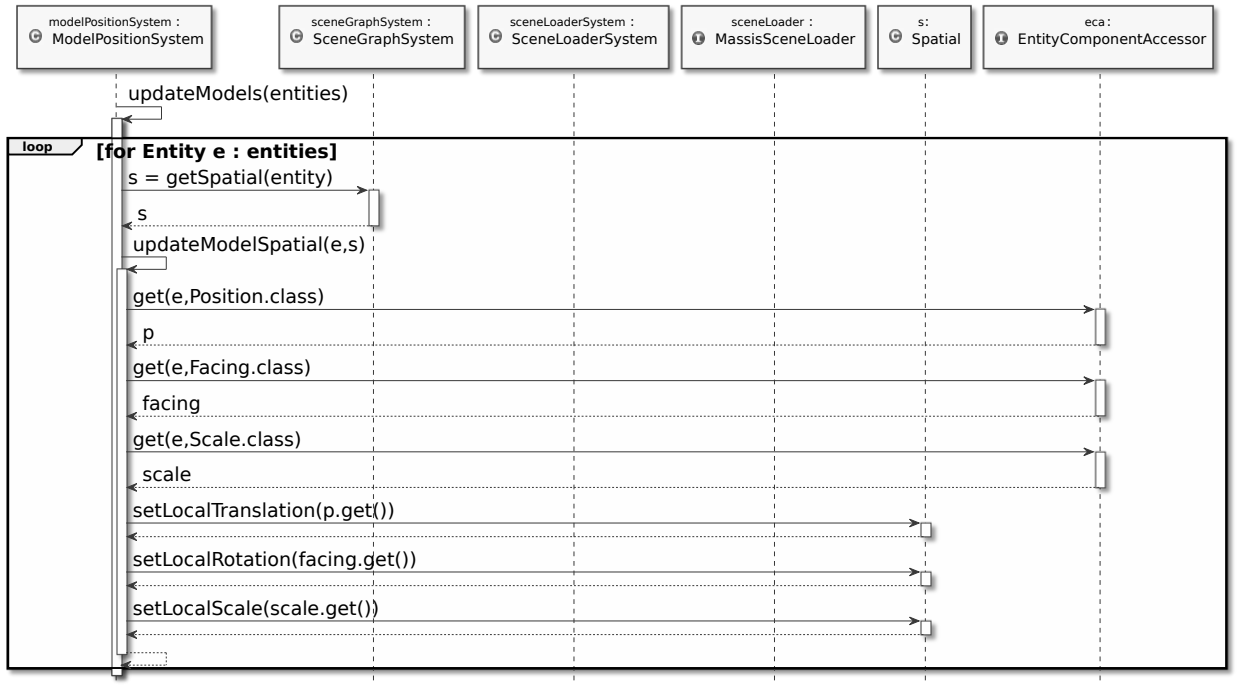

Figure 3.9: *Model Position System update sequence diagram*

AnimationSystem

Tracks entities with **animation components (AnimationComponent) and capable of being rendered (component Model3Dinfo)**, and performs all the necessary operations for applying them in the scene graph. Although the need for the `Model3DInfo` might seem redundant (an entity having an `AnimationComponent` always will have a `Model3DInfo` component), it has been considered a good idea to require entities to have also a `Model3DInfo`, as avoids errors (such as an Spatial not found). The main relationships of this system are shown in figure 3.10. The entities having an `AnimationComponent` are processed by this system, which retrieves the JME's `Spatial` object associated with each entity, and attaches the corresponding `AnimControl`. `AnimControl` is part of the JME3 libraries, and its task is to animate the spatial to which is attached.

The animation reference is provided by its name, and the *SceneLoader* provided to the *LoaderSystem* loads the animation from the file system (or cache or the network, depends on

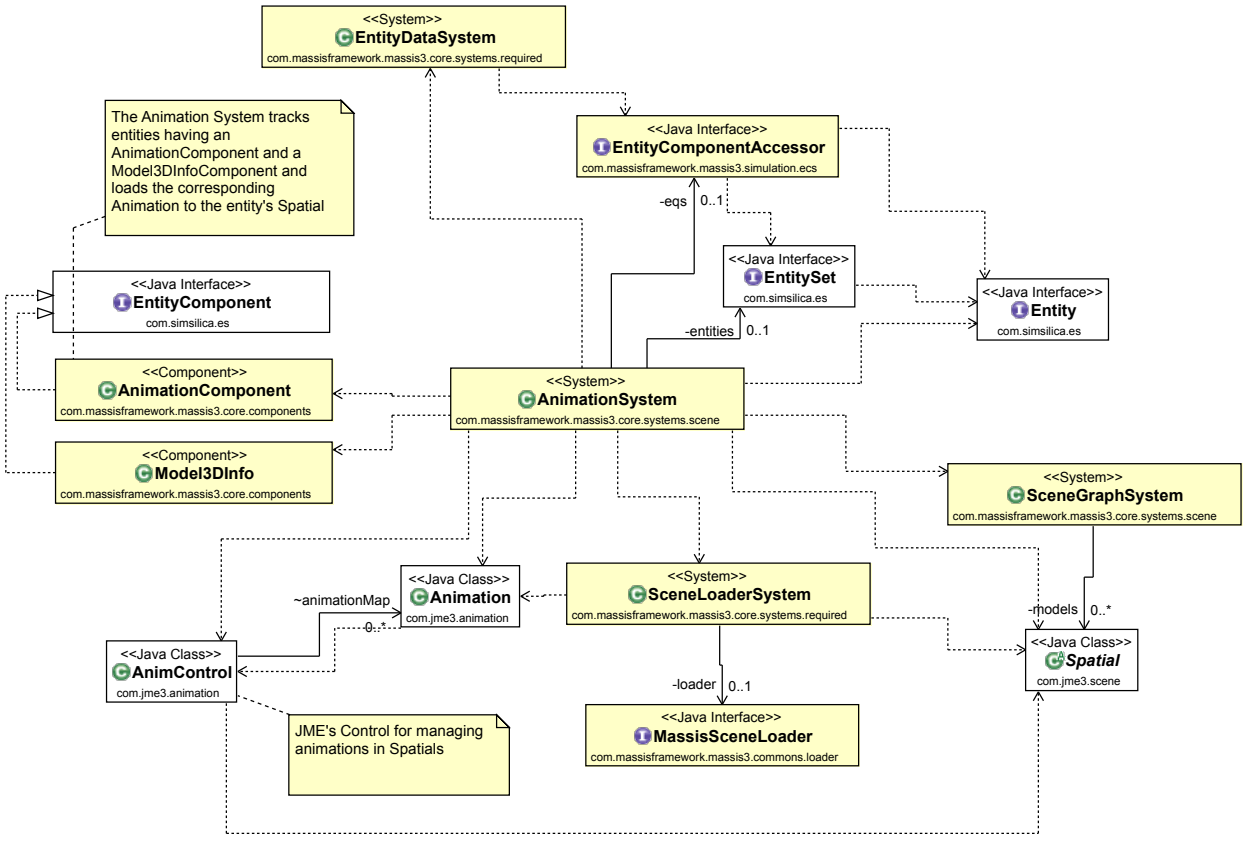

Figure 3.10: *Class diagram showing the relevant relationships of the Animation System*

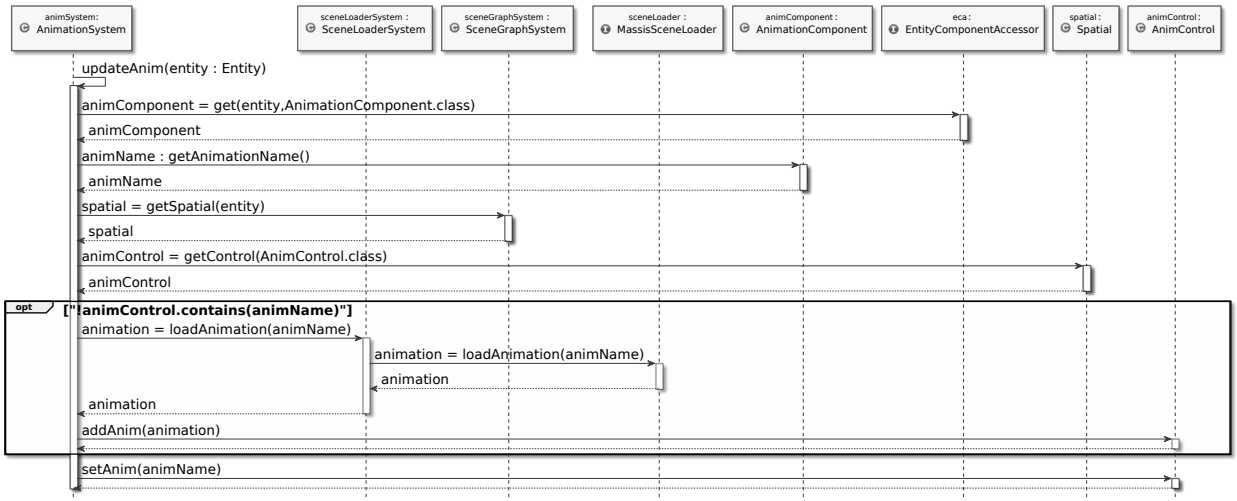

Figure 3.11: *diagram for the update of a single entity in the AnimationSystem*

its implementation) if necessary, and then the proper animation object is added to JME's AnimControl, and started in a *AnimationChannel*.

A simplification of this process is illustrated in figure 3.11. For those entities having an animation component, the corresponding necessary JME3 elements are retrieved, in this case a *Spatial* instance and its associated control. Then, for each control, the animation is loaded and enacted in JME3 only if it was not already active.

3.2.2 Scene Loader System

This system is in charge of loading different types of assets, that would be used in other systems. The task of loading complete scenes, animations, character, navigation meshes... is performed by this system. It is not associated with particular entities, so it does not look for components. It just facilitates access to animations, textures, and the like.

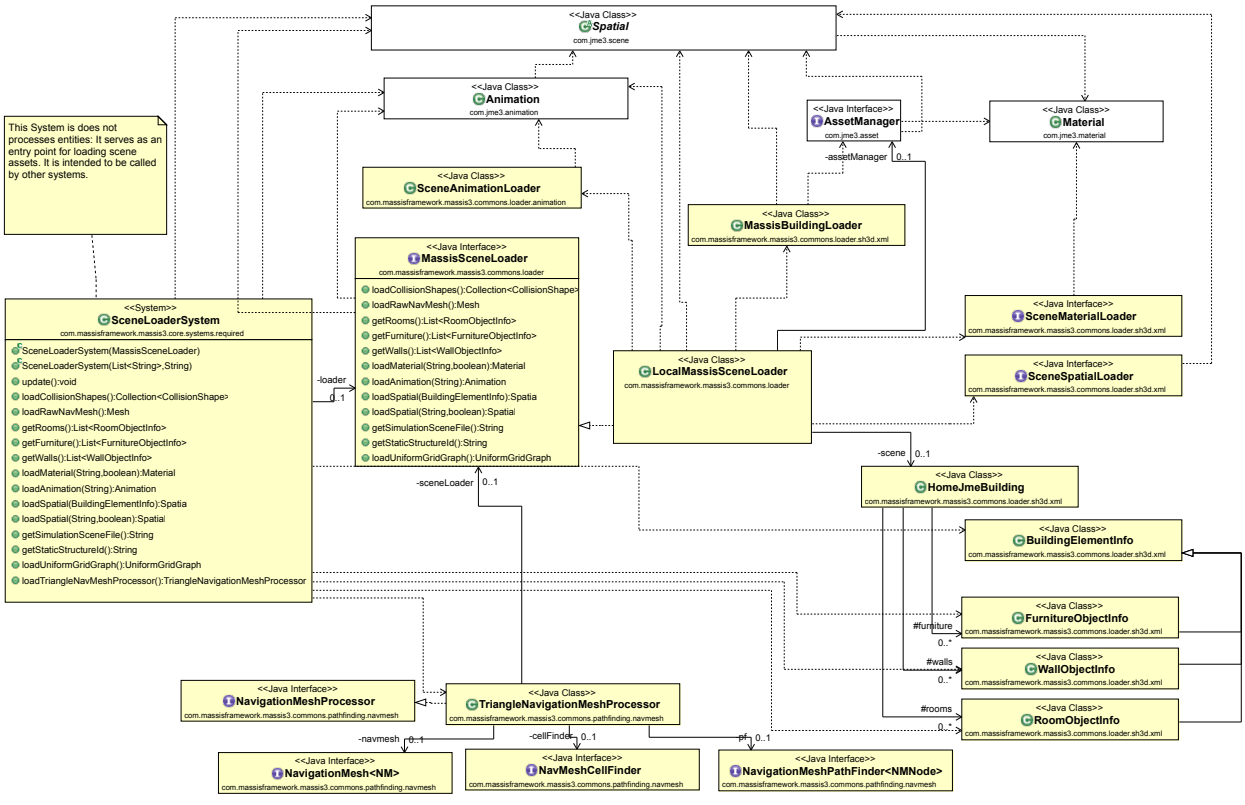

Figure 3.12: *Scene loader class diagram*

This is a very convenient way for maintaining consistency in the asset management tasks, and simplifies the access to resources by other systems. Whenever a System needs to access any kind of resource, should do it via this system. Figure 3.12 shows the relationships of this system in a class diagram.

3.2.3 Navigation Systems

The navigation system is an important part of the simulator. It fulfills some of the goals pointed out in the introduction related to the movement of the characters across the scene, i.e., the crowd simulation part. The navigation in general requires to locate the position of the character in the current building. Absolute coordinates are not valid, since the path is defined with respect to the floor of the character. Once located, the path generator system computes the navigation path along the ground. To adjust the displacement of the character with respect the animation and the update loop frequency, RVO2 system computes the velocity adjustments for the character to avoid collisions with other incoming character. When a collision is imminent with a obstacle (maybe some object moved or one character is coming against another), steering direction system intervenes to make one of the characters to change direction. Steering and velocity adjustments are stored as components associated to the entity. Collision Free Velocity System takes these adjustments to obtain the final velocity to associate to the agent. The next point to visit after a collision has been avoided is computed by the Clear Path Follow System.

Cell Location System & Path Generator System

The cell location system is in charge of keeping track of the current *navmesh cell* where an entity is. It tracks entities having the **Position** component. This is very useful for pathfinding tasks, because whenever an entity moves, it checks if the entity position is inside the polygon formed by the navmesh cell.

If its not, tries to find the nearest navmesh cell to the point of the entity. Although this might appear an expensive, unnecesary and time consuming computation (compute point-

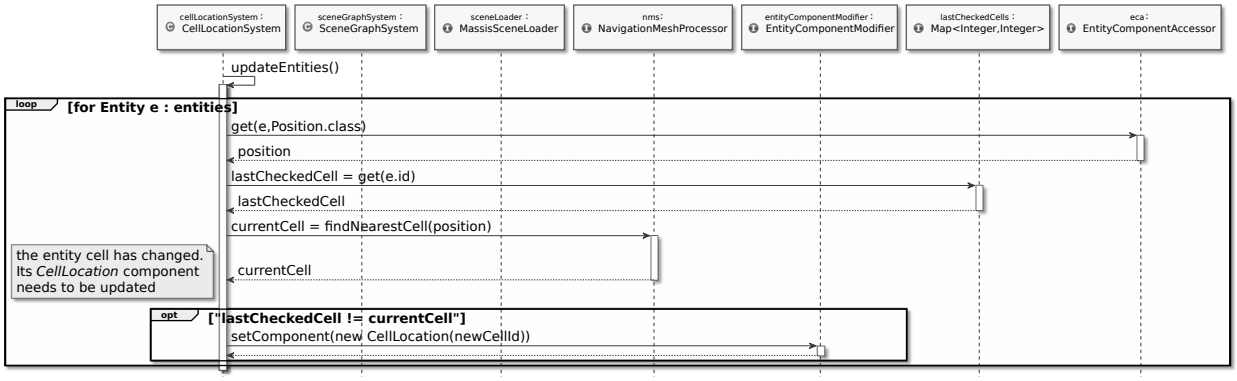

Figure 3.13: *Cell location system sequence diagram*

in-polygon algorithms every step), it is not: as entities trajectories are continuous almost every time (usually moving entities are virtual human avatars), the cells to be checked are at most 4 (in the case of triangular navmeshes) and 9 (in the case of square-based navmeshes): the last-known cell, and the cells surrounding it. The process executed by this system is illustrated in figure 3.13.

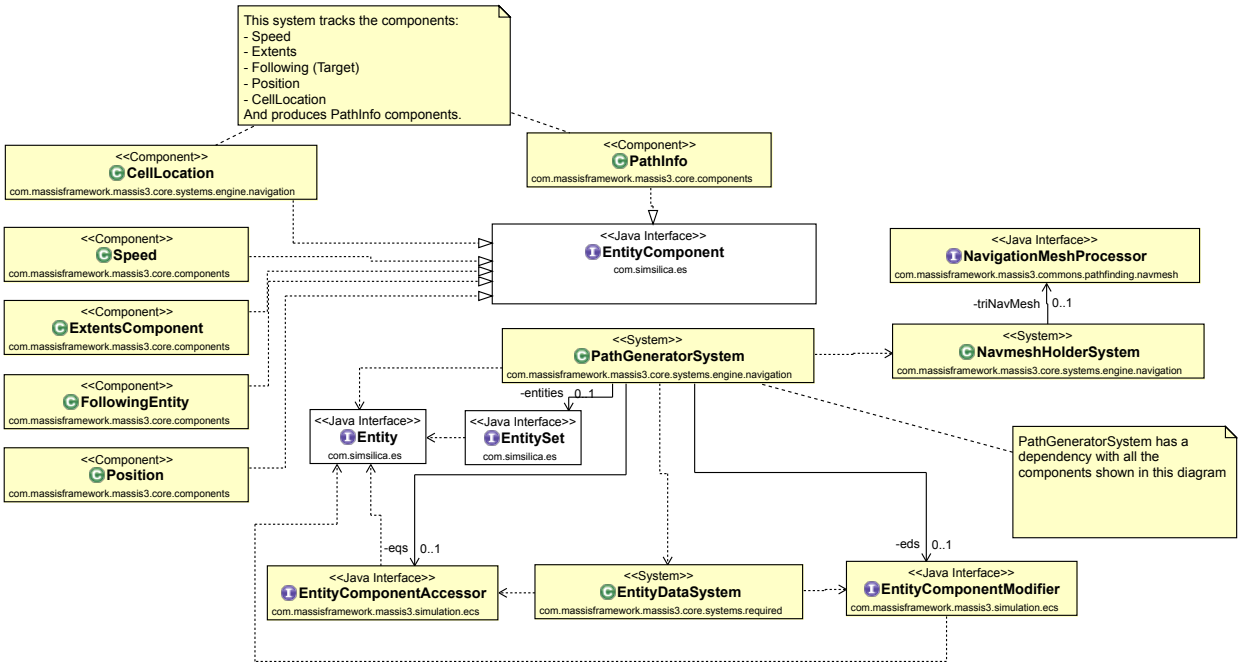

Figure 3.14: *Path Generator System Class diagram*

The Path Generator System is in charge of computing the path that agents in the

crowd should follow. It tracks entities with the **speed**, **Extends**, **Following**, and **Position** compoments (see figure 3.14). As it has been explained in section 2.1, pathfinding algorithms use some kind of graph representation of the environment under the hood. Thanks to the *Cell Location System*, which assigns a `CellLocationComponent` to every entity having a position component, this task is easy. The target location is also considered an entity, so it will also have a cell location component. The path generated by the pathfinding algorithm is then stored in the entity as a `PathInfo` component, which will be processed by other systems for executing other necessary movement computations (such as dynamic obstacle avoidance or moving the entity).

Direction Systems

Once the path has been computed, the next step is to make the virtual avatar follow that path. The path could be followed in different ways: directly or smoothed. The original path generated by the path generator system is a polygonal, unnatural path, because it relies on the navigation mesh structure (polygonal by design).

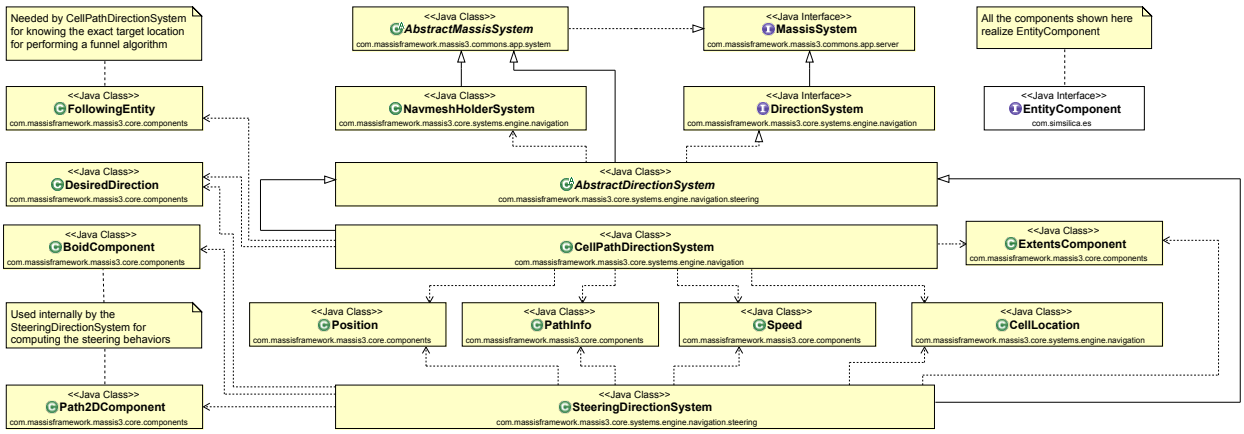

Figure 3.15: *Direction Systems Class diagram*

In order to have a more natural path, the `PathInfo` component must be processed, and a `Direction` should be generated by a *Direction System*. Currently, there are two direction systems implemented: The *Steering System* and the *CellPathDirectionSystem*.

These systems are *incompatible*: an entity cannot be processed by both systems at the same time. Figure 3.15 shows the relevant relationships between the components used by these systems.

Steering System The steering system aims to provide smoother, natural movements. It uses a velocity-based method for avoiding collisions documented in *Steering behaviors* [61]. It alters the trajectory of the character pretending the objects to avoid have mass that repels the character. The parameters needed are the agent's mass m, the agent's location $\vec{L}$, a maximum force f_{max} and a maximum speed s_{max}. Every step n, the computed steering forces are applied to the agent's location (limited by f_{max}), producing an acceleration whose magnitude is inversely proportional to the vehicle's mass.

$$\vec{A}_n = \left(\frac{trunc(\vec{F}_n, f_{max})}{m} \right) \tag{3.1}$$

The velocity of the agent in every step n is approximated by the Euler integration. Adding the velocity at the previous step ($\vec{V}_{n-1}$) to the current acceleration($\vec{A}_n$), produces a new velocity:

$$\vec{V}_n = trunc(\vec{V}_{n-1} + \vec{A}_n, s_{max}) \tag{3.2}$$

Finally, the velocity is added to the agent's location. These operations are shown graphically on figure 3.16.

$$\vec{L}_n = (\vec{L}_{n-1} + \vec{V}_n) \tag{3.3}$$

CellPathDirectionSystem Just performs a funnel algorithm [92], over the path generated and takes the first segment of the path as direction.

RVO2 System

The concept of a *Reciprocal Velocity Obstacle* (RVO) represents the velocity to associate to one character in the simulation to avoid a collision assuming that the other character

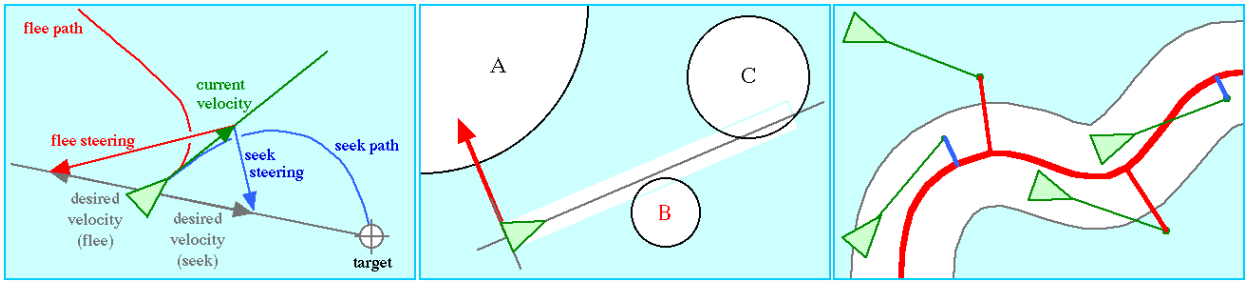

Figure 3.16: *Steering behaviors illustration*

will do likewise. This system computes RVO and stores into each moving entity. This system tracks entities having the components **RVO2Compoment**, **Speed**, **Position**. The computed RVO in each simulation frame is an average of the current velocity of the agent and the velocity lying outside the velocity obstacle of other agent. This concept can be formalized as:

$$RVO_B^A(\vec{v}_B, \vec{v}_A) = \left\{ \vec{v'}_A \mid 2\vec{v'}_A - \vec{v}_A \in VO_B^A(\vec{v}_B) \right\} \tag{3.4}$$

All the velocities for agent A being the average of the current velocity of the agent A ($\vec{v}_A$), and the velocity inside $VO_B^A(\vec{v}_B)$ (velocity obstacle of agent B) are contained in $RVO_B^A(\vec{v}_B, \vec{v}_A)$ (reciprocal VO of agent B to agent A). Figure 3.17. illustrates this equation. In Massis3, the RVO algorithm is provided by an external library, developed at University of North Carolina at Chapel Hill, and it is available via github [93]. Some modifications have been done to this library in order to make it usable in the framework developed, being the most important the adaptation to a 3D environment.

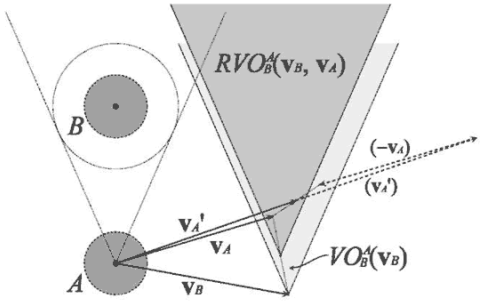

Figure 3.17: *The reciprocal Velocity obstacle $RVO_B^A(\vec{v}_B, \vec{v}_A)$ of agent B to agent A*

Figure 3.18 shows the relationships of the RVO2 System. The aspects of the entities that this system needs to know are the parameters for executing the RVO algorithm, (`RVO2Component`), and the agent position and its speed. With those components, the rvo2 algorithm can be executed, and the result of those operations (a new velocity), is stored in another component, `RVO2DesiredDirection`. This component would be further managed by another system, that tracks `RVO2DesiredDirection`s and computes new operations based on the rvo new velocity value.

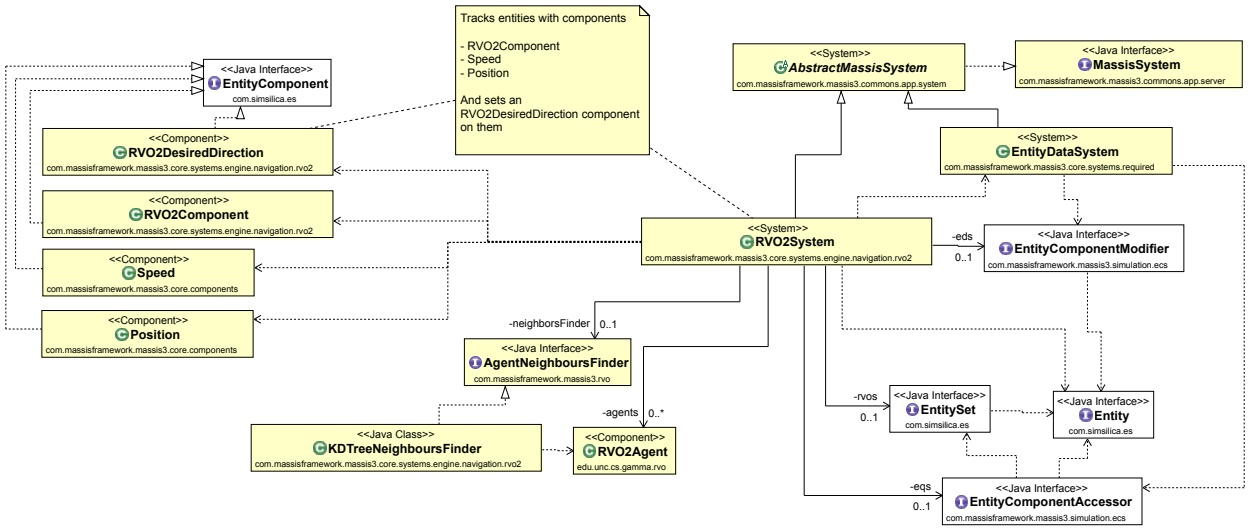

Figure 3.18: *Class diagram of the RVO2 System*

This system ensures that virtual humans do not bump with each other. Even the direction produced by the direction systems is incorrect, the application of the RVO2 algorithm ensures a collision free movement.

Collision Free Velocity Generator System

The different obstacle avoidance systems generate different *Direction* components, such as RVO2DesiredDirection. For obtaining the final, updated velocity of the agent, the value of the components produced by these systems is combined into a new component, *CollisionFreeVelocity*.

Clear Path Follow System

Once the *CollisionFreeVelocity* of an entity has been computed, it is time to compute the value of the new position of the entity. The required components for that computation are *Position*(the current location of the agent),*CollisionFreeVelocity* (the updated velocity value), and the nearest point on the floor, *NearestPointOnFloor* (which is computed by the GravitySystem).

Chapter 4

Scene and character design

The visual realism is attained by the rendering mechanisms of the game engine. However, for the sake of this work, some minor contributions have been necessary to successfully create the necessary elements to create the crowd simulations. These advances were used in the work [29] to improve the realism of the award winning demo.

A first element is the environment design tool. JME3 does not have any, and it needs to import meshes and textures from other tools which are not very user friendly. In this work, an import procedure from SweetHome3D editor was created to facilitate this task. The procedure is presented in 4.1.

The second element is the character animation. JME3 does not have a collection of characters and gestures and relies as well in external tools, not always directly compatible with JME3 formats. MakeHuman is a specialized tool prepared to create character animations. It is open-source and, as another contribution, a procedure was devised to import the characters into JME3. The procedure is presented in 4.2.

4.1 Design Tools

Spaces design plays an important role when it comes to crowd simulations. Involves the design of large areas, that should be close to reality. Designing such scenes is a difficult task, and often requires professional training. Also, most of them are not open-source, neither are free. Autodesk's software tools are a good example of non-free design tools: 3DMax [94],

Maya [95] or HomeStyler [96]. Blender [97] is an open source, multi platform and free alternative to 3D design. Also, JMonkey can load blender assets natively. However, the process of designing buildings with blender is still difficult and requires time and experience. Its graphical interface is shown in figure 4.2.

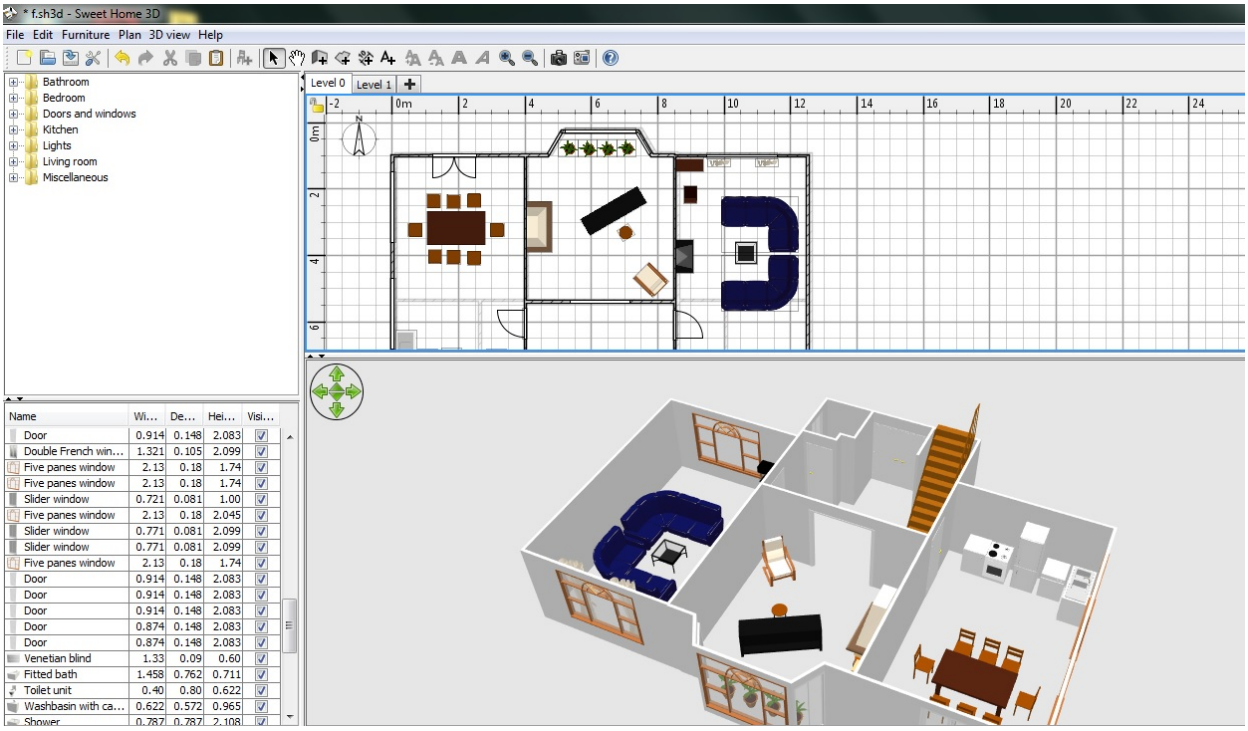

Figure 4.1: *SweetHome3D design software screenshot*

In Massis3, the chosen tool for designing spaces is SweetHome3D [98]. SweetHome3D is a well known package that is used to model all components involved in an indoor environment, such as walls, doors, stairs, people, etc. It is a free building design software application that allows users to create 3D houses in an easy way (its graphical interface is shown in figure 4.1), with 3D and 2D views. It also allows the decoration of the interior and the exterior of the building with a high level of detail: 3D objects can be imported as furniture, and they can be arranged for creating a virtual environment, which appearance can be improved adding multiple textures and light points. This software package has been used successfully in other simulation tools [99–102], due to its extensibility (it is open-source and can be extended via

plugins).

Figure 4.2: *Building design with blender*

During the development of this work, contributions to sweethome3D have been made, and this contributions have attracted the attention of the sweethome3d community, such as forum comments, feature requests and interviews to Rafael Pax [103]

4.1.1 Custom Scene Format

Although the conversion from SweetHome3D to other, well known 3D formats, such as OBJ [104], the conversion from other 3D formats (such as blender's) to sweethome3D is not possible without the loose of information (such as wall, room and furniture metadata).

In the development of Massis3, a custom scene format was developed. That way, the asset loading task becomes easier and more efficient. The specification of the scene format is open, JSON [105] based so tools can be implemented in order to convert other 3D formats to Massis3 scene format. This internal scene format consists on a zip file, containing the material definitions, texture files and j3o (JME binary format) files of the scene. The

meaning and content of these files is given by a description file, `scene.massis`, which contains the basic information of the building (rooms, doors, stairs, windows, furniture...), but not the detailed 3D information (e.g mesh triangles). Listing 4.1 shows (partially) an example content of the file.

```
{
 "version": "5200",
 "name": "Faculty1.sh3d",
 "sha1": "9deb18da70ac9297748af7e219a6faac8d0ea663",
 "modelsFile": "models_ed7361f1-37d7-41c7-99f2-2c44465fb647_.j3o",
 "materialsFolder": "materials_aa239d37-a49e-4363-9480-1ebeddc9c94e",
 "texturesFolder": "textures_eacb2ffe-9b99-4149-9a26-f42fb91143fa",
 "rooms": [
  {
   "name": "RoomSB1_D",
   "points": [
    ...
   ]
   ...
  }
 ]
}
```

Listing 4.1: *Example of an scene.json file contents*

4.2 Characters Design

Modeling virtual characters is a difficult and expensive task. Modeling a human character correctly requires a deep knowledge of specific applications and tools, and a significant amount of time. For most video-games or other applications, the number of different characters is limited, but when it comes to crowd simulation, this number should be as larger as possible. Although character modeling can be automatically done using 3D scanning technologies [106], the process often require expert knowledge for the operation. This issue can be solved using tools that speed up the design process, using predefined models, allowing performing variations to base models, such as height, weight, sex,age, etc. MakeHuman [107] is an open-source tool that facilitates this task. It comes with a friendly user interface (see figure 4.3) which makes easy the application of human body modification algorithms [108].

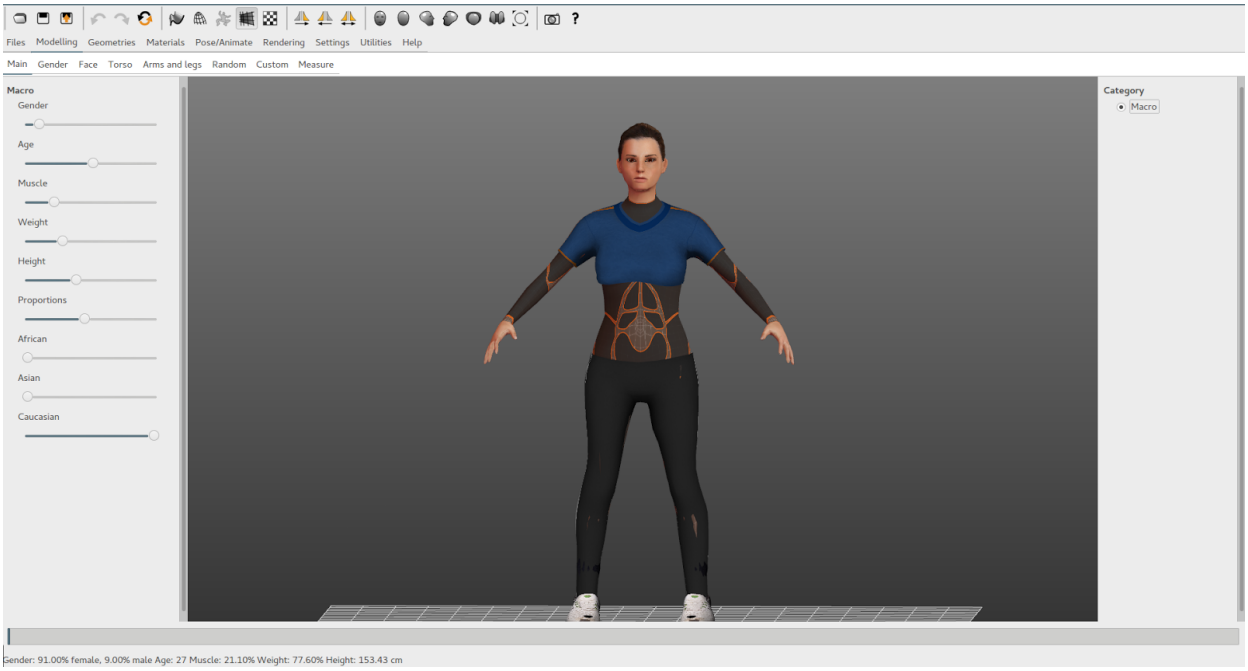

Figure 4.3: *MakeHuman IDE*

But the JME libraries do not support loading makehuman (`.mhx2`) files, but support loading blender (`.blend`) 3D models, and there is a plugin for blender that enables loading this kind of models into blender, so a set of scripts have been developed for making this step automatically (mhx2 -> blender -> j3o).

4.2.1 Characters animation

Character animation is a must when simulating realistic crowds. However, the process for animating virtual characters is not an easy task and requires additional 3D modeling knowledge and experience. A set of python extensions for blender have been developed, and packed into a Docker [109] container. The reasons for using Docker is that the extensions are dependant on the blender version. These extensions automatize the tasks of animation bone retargeting. These scripts are intended to work with Mixamo's [110] animations (the animation file format should be a Mixamo-style BVH skeleton)

Chapter 5

Case Study: Replicating crowd movement within a building

The case study reproduces a experiment made within the MOSI-AGIL project to determine the crowd simulation capabilities and how to infer behavior of larger groups of people. The experiment started with a pedestrian traffic data acquisition made by members of the project during two months in two buildings, the faculty of computer science and the faculty of Social Sciences and Politics of Universidad Complutense de Madrid.

Obtained data was used to filter possible populations and behaviors of groups of people. Given the crowd simulator, the goal was to evaluate if a simulation could be devised such as it produced similar pedestrian data as a result of the movement of the people. For this, the concept of *control zone* is defined. It is an area where traffic observations have been made. The traffic can be incoming or outgoing.

There are multiple uses for populations that adhere to the observed data:

- To evaluate hypothesis about the crowd behavior in the real world assuming the simulation implements in the navigation and activity functionality assumptions on the expected behavior of the pedestrians.
- If the behavior is accurate:
  - to extrapolate what are the pedestrians doing into other parts of the building.

– To infer too how many individuals are in the facility without needing to monitor other areas.

The results presented here have been published in press. The algorithm was published in [30]. The role of the crowd simulation activity within the smart environment design was published in [28]. An evolution of this idea was published in [32] to enable the development of intelligent sensors. Part of the published content is reproduced in this section to show how the architecture from section 3.2 was used. The papers have been included as annexes to be consulted.

5.1 Observed data

The simulation is correct as long as it shows the same amount of traffic in control zones as a previously recollected data. Collected data is the incoming/outgoing traffic data observed within some control zones. These concepts are defined below:

Control zone

We define a *control zone* C as any zone in the environment where the crowd density has been measured over time. Normally, it is a room, a corridor, or an area around a door. Figure 5.4 illustrates a set of control points placed in an environment.

Time record

The number of people that have entered a control zone C_i during a concrete time interval is represented by a *time record*, which consists on a triple $T_i = \{t, n, C_i\}$, where n is the number of people that have crossed the control zone in the time interval $t = [t_{start}, t_{end}]$. The value of t is expressed as an interval because depending on the people counting method, this data cannot be measured with exact precision (people counted by a human observer, for example).

Time interval	FS_0	FS_1	FS_2	FS_3
$[0, 60)$	6	7	5	3
$[60, 120)$	5	3	1	5
$[120, 180)$	6	1	6	9
$[180, 240)$	1	0	2	1
$[240, 300)$	4	-	3	8
$[300, 360)$	4	-	1	6
$[360, 420)$	5	-	3	4
$[420, 480)$	6	-	5	3
$[480, 540)$	5	-	1	5
$[540, 600)$	6	-	6	9
$[600, 660)$	1	0	2	1
$[660, 720)$	4	0	3	8

Table 5.1: *Example of a flow table*

Flow set

All the time records belonging to a particular control zone C_i are grouped in a flow set $FS_i = \{T_{i_0}, T_{i_1}, \ldots, T_{i_n}\}$. The natural way for representing this data is arranging the time records in a table, (called *flow table*) being in the same row the ones having the time intervals in common. Table 5.1 illustrates how this arrangement is done. An entry containing a hyphen (-) instead of a number means that there is no data available for that time interval in the control zone (perhaps it was collected with different time intervals, or because the observation of the people flow was not performed during that time).

5.2 Crowd simulation population setup

Preparing the crowd includes determining how many people is involved and what they can be doing while traversing the facility. No particular activities of the daily living are considered and, for the sake of the experiment, the behavior of a character in the simulation will be summarized as a sequence of transitions from one place to another, perhaps making stops between transitions. The stop actions will represent to some extent the activities performed in the building.

It is not known what precise navigation paths each individual followed. As a result, many possible populations may fit the observed data. To answer this question, a greedy algorithm was devised and presented in [30].

This approach creates trajectories passing through the control zones, taking into account the distances between them and the speed of each agent. It is assumed that there are no agents at the beginning of the simulation, that they come from outside of the environment via some specified *entrance points*, and that they exit the environment through another entrance point. The trajectories are computed in a reverse way: The first point added to the trajectory is the last one the agent will pass through. At each iteration, a set of reachable control zones is selected by a *selection function*, which takes into account the agent's speed and the the current control zone of the agent. Then, the number of people in the selected control zone is decreased by 1. The process is repeated until the first time interval is reached.

Listing 1 Main Algorithm

```
while Flow table is not empty do
    endLoc ← selectEntrance()
    tr ⟵ the time record with maximum time
    tr.n = tr.n − 1
    availableTime ← tr.t
    loc ← tr.c
    agentSpeed ← speedSelection()
    availableTime ← availableTime − travelTime(endLocation, loc)
    trace ← GenerateTrace(availableTime, loc, tr)
    traces ← traces + GenerateTrace(availableTime, loc, tr, speed)
end while
    return traces
```

The algorithm ends when the flow table is empty: If the n value of all the time records is zero, there is no need to generate trajectories.

At the beginning, an *entrance point* is selected. It will be the location where the agent will *exit* from the environment (remember that the algorithm runs in a reverse way). After that, a non-empty time record (tr) with the highest time value is selected. The control

zone corresponding to this time record will be the *last* control zone that the agent will cross before exiting the environment. If there are more than one available, can be chosen with a custom criteria.

Listing 2 Trace Generation

```
trace ← INITIALIZETRACE
while availableTime ≥ 0 do
    available ← AVAILABLETARGETS(availableTime, loc)
    if available is not empty then
        tr ← SELECTTARGET(available)
        availableTime ← availableTime − TRAVELTIME(c, loc, speed)
        loc ← LOCATIONOF(tr)
        tr.n = tr.n − 1
        ADDTOTRACE(loc)
    else
        if availableTime > 0 then
            ip = SELECTINTERMEDIARYPOINT()
            availableTime ← availableTime − TRAVELTIME(ip, loc, speed)
            ADDTOTRACE(loc)
        end if
        if availableTime ≤ 0 then
            go to the start point
            eP ← SELECTENTRANCE(B)
            ADDTOTRACE(loc, trace)
            availableTime ← availableTime − TRAVELTIME(eP, loc, speed)
        end if
    end if
end while
    return trace
```

The n value of the selected time record ($t.n$) is decremented by 1. This is done because if n agents will be passing through that the corresponding control zone in the time interval $t.t$. Doing this operation n times for different agents (or the same agent, if the time interval is large enough), will satisfy the condition of the initial problem. After the initialization, the process of selecting time records is done iteratively, applying the same concept:

- An available time record is selected. A time record tr is "available" if the $tr.n > 0$ and the agent can reach it in the time interval $t.t$.

- If there is no time record available (There are too far from the agent's location, for example), an intermediary point in the environment is selected.
- Every time that a location is selected, the available time is decreased, taking into account the distance from the agent location and the agent's speed.
- When the *available time* is less or equal to zero, an entrance point is selected. This entrance point will be the *first one* that the agent will cross.

5.3 Additional Systems involved

The simulation required the following additional systems. They were incorporated to have a better visual feedback of the simulation progress.

Figure 5.1: *Simulation visual output showing identifiers over human avatars (left) and graphical representation of the navigation mesh, target points and paths of the virtual humans (right).*

A system to draw an ID over each person (`ShowIdsDebugSystem`). The class diagram showing the relationships of this system with other simulation systems is shown in figure 5.2.

The purpose of this system is to facilitate the debugging of the simulation. By adding IDs, it is easier to identify characters whose behavior is not the adequate. Figure 5.1 shows the graphical output of this system.

A system to draw paths over the scene. Another debugging system that allows to foresee the results of the generated paths by the algorithm and confront them with the actual navigation path as generated by the navigation system. The output of this system can be seen in figure 5.1

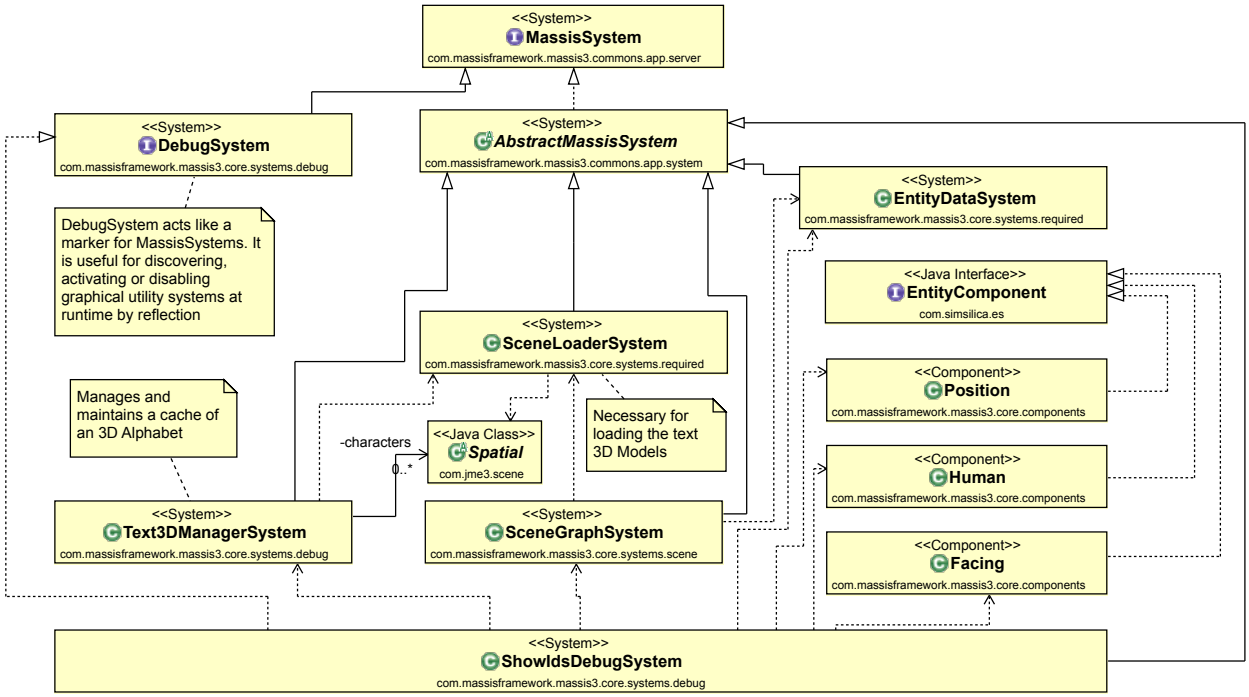

Figure 5.2: *Class diagram of the ShowIdsDebugSystem*

5.4 Example

For illustration purposes, let's consider a simple example with three control zones, C_0, C_1 and C_2. For an agent with an specific speed, the travel times in seconds between the control zones could be 200 ($C_0 \rightarrow C_1$), 100 ($C_0 \rightarrow C_2$) and 150 ($C_1 \rightarrow C_2$), as shown in Figure 5.3. The initial state of the flow table is shown in Table 5.2.

When the algorithm starts, a non-empty record having the time interval with the highest value is chosen. In this example the chosen time record shown in table 5.2. The candidate time records are the ones highlighted, and the chosen time record ($\{[300, 360), 4, C_0\}$) is marked with an arrow ($\leftarrow$).

Once the time record is selected, its value is reduced by one, and the location of the agent is saved. The candidate time records after the first step are highlighted in table 5.3. The time intervals are $[240, 300)$ and $[120, 180)$, because traveling from C_0 to C_2 takes 100 seconds, and $359 - 100 = 259 \in [240, 300)$, being the other case analogous. In this step, the candidate time record that will be chosen will be $\{[240, 300), 3, C_2\}$, because its time interval is higher.

In the next iteration, the candidate time records are $\{[60, 120), 3, C_1\}$ and $\{[120, 180), 6, C_0\}$.

If we follow the same process that the one done in the previous step, the selected time record should be $\{[120, 180), 6, C_0\}$. But it is obvious that the trajectories of the agents would be awkward, moving forward and backwards every step. This restriction depends on

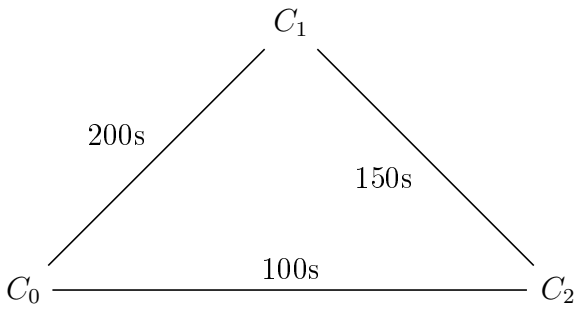

Figure 5.3: *Travel times from the different control zones of the example*

Time interval	FS_0	FS_1	FS_2
$[0, 60)$	6	7	5
$[60, 120)$	5	3	1
$[120, 180)$	6	1	6
$[180, 240)$	1	0	2
$[240, 300)$	4	5	3
$[300, 360)$	4 $\leftarrow$	0	1

Table 5.2: *Initial state.*

Time interval	FS_0	FS_1	FS_2
$[0, 60)$	6	7	5
$[60, 120)$	5	3	1
$[120, 180)$	6	1	6
$[180, 240)$	1	0	2
$[240, 300)$	4	5	3 $\leftarrow$
$[300, 360)$	[3]	0	1

Table 5.3: *Second iteration.*

Time interval	FS_0	FS_1	FS_2
$[0, 60)$	6	7	5
$[60, 120)$	5	3 $\leftarrow$	1
$[120, 180)$	6	1	6
$[180, 240)$	1	0	2
$[240, 300)$	4	5	[2]
$[300, 360)$	3	0	1

Table 5.4: *Third iteration.*

Time interval	FS_0	FS_1	FS_2
$[0, 60)$	6	7	5
$[60, 120)$	5	[2]	1
$[120, 180)$	6	1	6
$[180, 240)$	1	0	2
$[240, 300)$	4	5	2
$[300, 360)$	3	0	1

Table 5.5: *Last iteration*

how the control zone selection function is implemented.

The last step is shown in table 5.5. After this last step, a path from the control zone chosen (e.g., C_0 or C_2) to an entrance point should be computed.

5.5 Preparing the environment

The preparation of the environment was made with the design tools presented in section 4. The SweetHome3D software was used to generate a blueprint of the computer science faculty, whereas the MakeHuman allowed to produce different characters with different animations and sizes to make the simulation more credible.

The case study considers the floor of a building (see Figure 5.4), where control areas, painted in red, have been defined. Using this blueprint, samples of traffic data through red areas is used as input. The number of people crossing that control zones was annotated

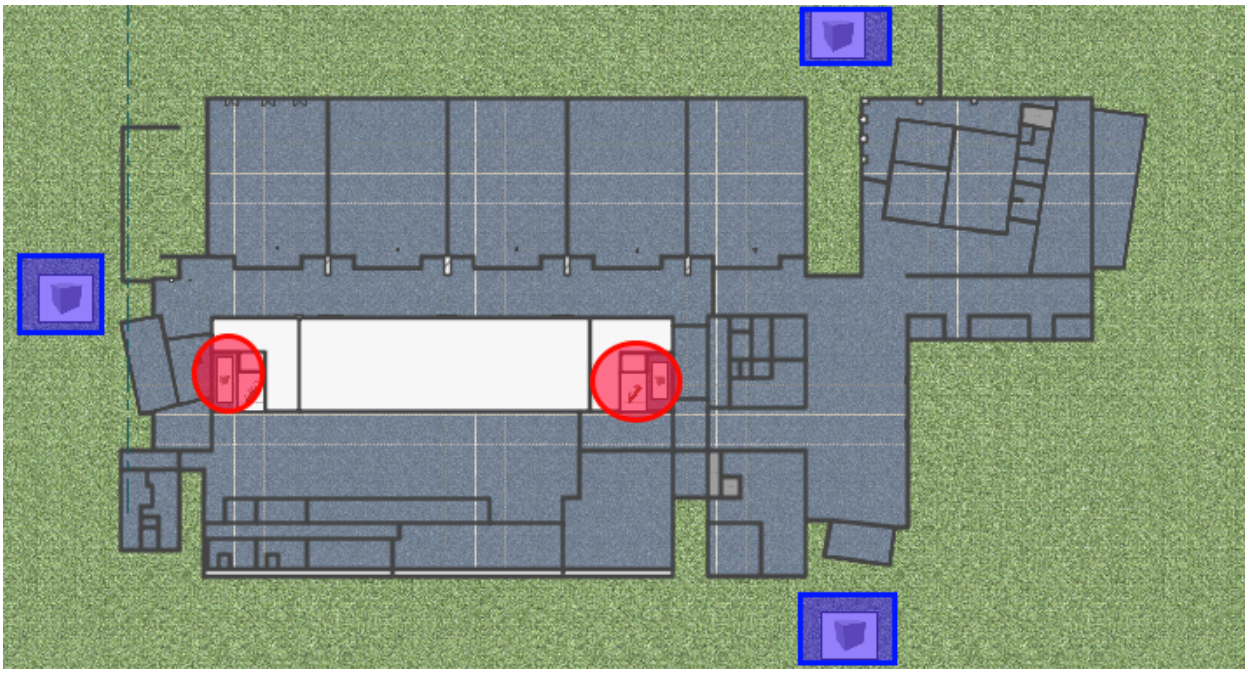

Figure 5.4: *Control zones of the case study.*

with a frequency of one minute.

The crowd behavior consists of entering the building through designed entrances and move along the floor in a way that traffic data through red areas matches input data.

Data used for the experiment comes from a real human traffic observation in a configuration similar to the one depicted in Figure 5.4, where the control zones are marked as red circles, and entrance points are marked as blue squares.

The traffic of the inhabitants during the simulation is represented in Figure 5.5. The characters move along different pathways automatically generated and depicted with the additional systems.

These pathways are designed in a way that, assuming a constant speed, the characters cross the intended red areas at the designated times. The pathways may look unnatural because of this limitation. Hence, one can observe a character going to the middle of a room and just returning.

One of the future work improvements consists in adding activities to perform along the

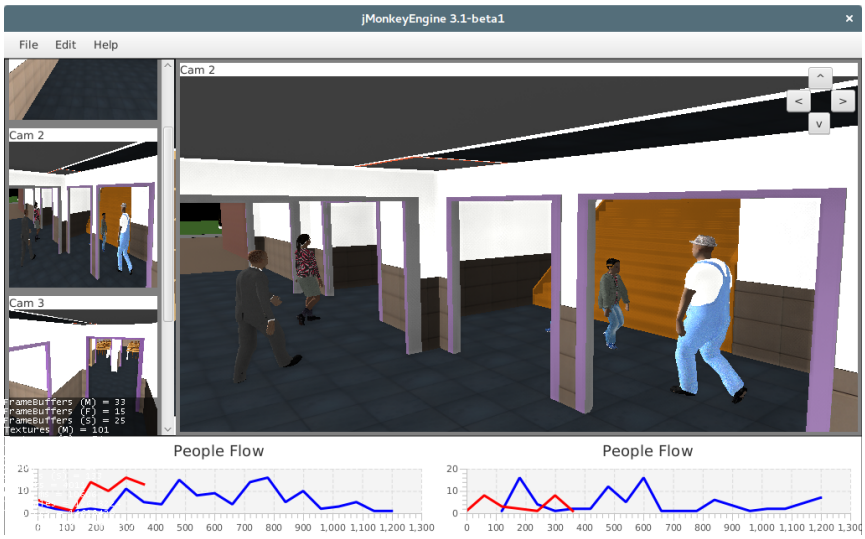

Figure 5.5: *Sample execution of the simulation over the blueprint from figure 5.4*

pathways so that the simulation is more realistic. Despite this limitation, the obtaining traffic data from the simulation is close to the ideal.

5.6 Simulation results

Time interval	Real	Simulated
0	4	10
60	1	4
120	4	9
180	7	9
240	2	6
300	8	14

Table 5.6: *Control zone A density values*

Time interval	Real	Simulated
0	10	14
60	1	2
120	6	9
180	1	5
240	5	12
300	2	3

Table 5.7: *Control zone B density values*

Some of the observed results during one execution are presented in figure 5.6 and in tables 5.6 and 5.7. It can be observed a difference between the number of people that passed over the control zone in the real measurements and the simulated ones. Two reasons may explain these results. First, the algorithm does not take into account (yet) the movements of

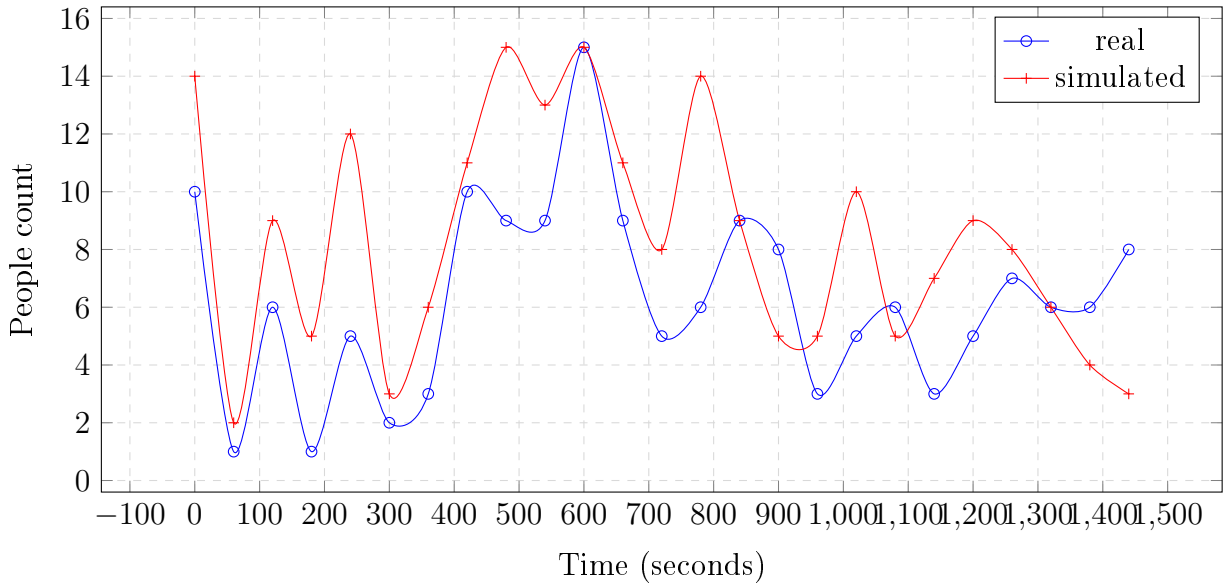

Figure 5.6: *Control zone crowd density values, real vs simulated.*

other agents during its execution, the collision detection between agents and its resolution is done during the simulation, not before. Second, a trajectory may cross an undesired control zone, making the results slightly worse. The plot reported in Figure 5.6 shows how this issue alters the simulation results, increasing the number of people counted over one of the control points of the simulation.

Chapter 6

Conclusions & Future work

This work has presented a framework for simulating crowds in large environments. The framework was built over the JMonkey game engine (JME3) and used textures, meshes and animations imported from existing artwork repositories. Extensions have been made for third party design tools to make easier the environment and human avatar modeling task. The framework implements an Entity Component System pattern that facilitates the experimentation with crowd simulation problems. Known algorithms for steering and path smoothing were implemented using as reference research papers. It also integrates libraries with implemented algorithms for collision avoidance. The system was constantly expanded with different systems proving its capability for evolving and as platform for experimenting with crowd simulation algorithms. As a simulation tool is more versatile than existing alternatives based on closed source game engines.

The system is a complete reconstruction of previous work published as the MASSIS framework. Fundamental defects in the simple rendering engines MASSIS used discouraged its use for further development. The decision proved to be adequate since the results are comparable to professional tools in terms of visual quality.

The framework has proven itself successful through several papers that contain results related to preliminary work made with MASSIS and its evolution with JMonkey, which is the core of this contribution. It has contributed to the scientific community with several conference papers, and two journal articles.

The results of this work have been published in part. The preliminary attempts to use MASSIS for this work were published in [26] and [27]. The case study was published in [30]. The application of the methodology for simulating crowds in cooperation with social scientists was published in [28] and [31]. The benefits of exploting the data produced by the simulations was published in [29] and [32].

There is still a lot of work to be done. More crowd behavior algorithms can be added in order to make simulations better, and rendering methods could be improved in order to make simulations more realistic visually.

But executing the simulations on the researcher or final client's computer limits the possibilities that a simulation can offer. A good approach for solving this issue is to execute the simulations in the cloud.

During the last decade, the cloud computing paradigm [111] has demonstrated to be an effective way for the organization and deployment of software architectures. Aiming to provide functionalities *as a service*, this approach has been used for the development of multiple types of solutions, being the most known SaaS (*Software as a Service*) [112], PaaS(*Platform as a Service*) [113], or IaaS(*Infrastructure as a Service*) [114].

Allowing remote access to the simulation via different methods and protocols and standards can alleviate the performance problems that arise when running large scale, 3D crowd simulations in commodity hardware, and opens the possibility of the integration of a running simulation as a group of services (*microservices*) that could be consumed by third parties.

The next step in the development of the framework will be extending the simulator framework to be able to be able to execute simulations on the cloud.

Bibliography

[1] Mark Weiser, Rich Gold, and John Seely Brown. The origins of ubiquitous computing research at parc in the late 1980s. *IBM systems journal*, 38(4):693–696, 1999.

[2] Paddy Nixon, Simon Dobson, and Gerard Lacey. Managing smart environments. In *Proceedings of the Workshop on Software Engineering for Wearable and Pervasive Computing*, 2000.

[3] Diane Cook and Sajal Das. *Smart environments: Technology, protocols and applications*, volume 43. John Wiley & Sons, 2004.

[4] G Michael Youngblood, Edwin O Heierman, Lawrence B Holder, and Diane J Cook. Automation intelligence for the smart environment. In *International Joint Conference On Artificial Intelligence*, volume 19, page 1513. LAWRENCE ERLBAUM ASSOCIATES LTD, 2005.

[5] Bill Von Neida, Dorene Manicria, and Allan Tweed. An analysis of the energy and cost savings potential of occupancy sensors for commercial lighting systems. *Journal of the Illuminating Engineering Society*, 30(2):111–125, 2001.

[6] Marco Jahn, Marc Jentsch, Christian R Prause, Ferry Pramudianto, Amro Al-Akkad, and Rene Reiners. The energy aware smart home. In *Future Information Technology (FutureTech), 2010 5th International Conference on*, pages 1–8. IEEE, 2010.

[7] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, pages 1–6. ACM, 2010.

[8] Claudio Martani, David Lee, Prudence Robinson, Rex Britter, and Carlo Ratti. Enernet: Studying the dynamic relationship between building occupancy and energy consumption. *Energy and Buildings*, 47:584–591, 2012.

[9] Irfan A Essa. Ubiquitous sensing for smart and aware environments. *IEEE personal communications*, 7(5):47–49, 2000.

[10] Dairazalia Sánchez, Monica Tentori, and Jesús Favela. Activity recognition for the smart hospital. *IEEE intelligent systems*, 23(2), 2008.

[11] Susan Mckeever, Juan Ye, Lorcan Coyle, Chris Bleakley, and Simon Dobson. Activity recognition using temporal evidence theory. *Journal of Ambient Intelligence and Smart Environments*, 2(3):253–269, 2010.

[12] Mohamed Tarik Moutacalli, Abdenour Bouzouane, and Bruno Bouchard. Unsupervised activity recognition using temporal data mining. *Sensors*, 1:5214324683, 2012.

[13] Diane J Cook, Michael Youngblood, Edwin O Heierman III, Karthik Gopalratnam, Sira Rao, Andrey Litvin, and Farhan Khawaja. MavHome: An agent-based smart home. In *2013 IEEE international conference on pervasive computing and communications (PerCom)*, pages 521–521. IEEE Computer Society, 2003.

[14] Juan Carlos Augusto and Chris D Nugent. *Designing smart homes: the role of artificial intelligence*, volume 4008. Springer Science & Business Media, 2006.

[15] Sajal K Das and Diane J Cook. Designing smart environments: A paradigm based on learning and prediction. In *Pattern Recognition and Machine Intelligence*, pages 80–90. Springer, 2005.

[16] Andrea Omicini, Alessandro Ricci, and Giuseppe Vizzari. Building smart environments as agent workspaces. In *Enabling Technologies: Infrastructure for Collaborative*

Enterprises, 2007. WETICE 2007. 16th IEEE International Workshops on, pages 92–97. IEEE, 2007.

[17] HUBERT MRóZ and JAROSŁAW WĄS. Discrete vs. continuous approach in crowd dynamics modeling using gpu computing. *Cybernetics and Systems*, 45(1):25–38, 2014.

[18] Jeremy Shopf, Joshua Barczak, Christopher Oat, and Natalya Tatarchuk. March of the froblins: Simulation and rendering massive crowds of intelligent and detailed creatures on gpu. In *ACM SIGGRAPH 2008 Games*, SIGGRAPH '08, pages 52–101, New York, NY, USA, 2008. ACM.

[19] Ugo Erra, Rosario De Chiara, Vittorio Scarano, and Maurizio Tatafiore. Massive simulation using gpu of a distributed behavioral model of a flock with obstacle avoidance. *Proceedings of Vision, Modeling and Visualization 2004 (VMV)*, 2004.

[20] Dan Chen, Lizhe Wang, Xiaomin Wu, Jingying Chen, Samee U. Khan, Joanna Kołodziej, Mingwei Tian, Fang Huang, and Wangyang Liu. Hybrid modelling and simulation of huge crowd over a hierarchical grid architecture. *Future Generation Computer Systems*, 29(5):1309 – 1317, 2013. Special section: Hybrid Cloud Computing.

[21] Artur Malinowski, Paweł Czarnul, Krzysztof Czuryło, Maciej Maciejewski, and Paweł Skowron. Multi-agent large-scale parallel crowd simulation. *Procedia Computer Science*, 108:917 – 926, 2017. International Conference on Computational Science, {ICCS} 2017, 12-14 June 2017, Zurich, Switzerland.

[22] Daniel Thalmann, Helena Grillon, Jonathan Maim, and Barbara Yersin. Challenges in crowd simulation. In *CyberWorlds, 2009. CW'09. International Conference on*, pages 1–12. IEEE, 2009.

[23] Michal Ponder, George Papagiannakis, Tom Molet, Nadia Magnenat-Thalmann, and Daniel Thalmann. Vhd++ development framework: Towards extendible, component

based vr/ar simulation engine featuring advanced virtual character technologies. In *Computer Graphics International, 2003. Proceedings*, pages 96–104. IEEE, 2003.

[24] Mankyu Sung, Michael Gleicher, and Stephen Chenney. Scalable behaviors for crowd simulation. In *Computer Graphics Forum*, volume 23, pages 519–528. Wiley Online Library, 2004.

[25] Daniel Masamune Hall. Ecs game engine design. 2014.

[26] Rafael Pax and Juan Pavón. Agent-based simulation of crowds in indoor scenarios. In *Intelligent Distributed Computing IX*, pages 121–130. Springer, Cham, 2016.

[27] Rafael Pax and Juan Pavón. Agent architecture for crowd simulation in indoor environments. *Journal of Ambient Intelligence and Humanized Computing*, 8(2):205–212, 2017.

[28] Jorge J Gomez-Sanz, Rafael Pax, Millán Arroyo, and Marlon Cardenas Bonett. Requirement engineering activities in smart environments for large facilities. *Computer Science & Information Systems*, 14(1), 2017.

[29] Jorge J Gómez-Sanz, Marlon Cárdenas, Rafael Pax, and Pablo Campillo. Building prototypes through 3D simulations. In *Advances in Practical Applications of Scalable Multi-agent Systems. The PAAMS Collection: 14th International Conference, PAAMS 2016, Sevilla, Spain, June 1-3, 2016, Proceedings*, volume 9662, page 299. Springer, 2016.

[30] Rafael Pax and Jorge J Gómez-Sanz. A greedy algorithm for reproducing crowds. In *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection*, pages 287–296. Springer International Publishing, 2016.

[31] Jorge J Gomez-Sanz, Rafael Pax, and Millán Arroyo. Towards the simulation of large environments. In *AmILP@ ECAI*, 2016.

[32] Rafael Pax, Marlon Cárdenas Bonett, Jorge J Gómez-Sanz, and Juan Pavón. Virtual development of a presence sensor network using 3d simulations. In *International Conference on Smart Cities*, pages 154–163. Springer, Cham, 2017.

[33] Bobby Anguelov. *Video game pathfinding and improvements to discrete search on grid-based maps.* phdthesis, University of Pretoria, 2012.

[34] Vadim Bulitko, Yngvi Björnsson, and Ramon Lawrence. Case-based subgoaling in real-time heuristic search for video game pathfinding. *Journal of Artificial Intelligence Research*, 39(1):269–300, 2010.

[35] Changyun Wei, Koen V Hindriks, and Catholijn M Jonker. Multi-robot cooperative pathfinding: A decentralized approach. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 21–31. Springer, 2014.

[36] Harri Antikainen. Using the hierarchical pathfinding a* algorithm in GIS to find paths through rasters with nonuniform traversal cost. *ISPRS International Journal of Geo-Information*, 2(4):996–1014, 2013.

[37] S Umashankar and others. Optimization on shortest path finding for underground cable transmission lines routing using gis. *Journal of Theoretical & Applied Information Technology*, 65(3), 2014.

[38] Christine E Dunn and David Newton. Optimal routes in GIS and emergency planning applications. *Area*, pages 259–267, 1992.

[39] Zeyad Abd Algfoor, Mohd Shahrizal Sunar, and Hoshang Kolivand. A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology*, 2015:7, 2015.

[40] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[41] Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195:470 – 495, 2013.

[42] Hang Ma and Sven Koenig. Optimal target assignment and path finding for teams of agents. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 1144–1152. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[43] David Silver. Cooperative pathfinding. *AIIDE*, 1:117–122, 2005.

[44] Marika Ivanová and Pavel Surynek. Adversarial cooperative path-finding: A first view. In *AAAI (Late-Breaking Developments)*, 2013.

[45] Mikko Mononen. Navigation-mesh toolset for games. *GitHub Recast and Detour*, 2014.

[46] Peter Yap. *Grid-Based Path-Finding*, pages 44–55. Springer Berlin Heidelberg, 2002.

[47] Daniel Damir Harabor, Alban Grastien, et al. Online graph pruning for pathfinding on grid maps. In *AAAI*, 2011.

[48] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[49] Emilio Serrano and Juan Botia. Validating ambient intelligence based ubiquitous computing systems by means of artificial societies. *Information Sciences*, 222(0):3 – 24, 2013. Including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems.

[50] Ko-Hsin Cindy Wang, Adi Botea, et al. Fast and memory-efficient multi-agent pathfinding. In *ICAPS*, pages 380–387, 2008.

[51] Nuria Pelechano and Ali Malkawi. Evacuation simulation models: Challenges in modeling high rise building evacuation with cellular automata approaches. *Automation in construction*, 17(4):377–385, 2008.

[52] Rafael Pax Sánchez. Massis: Multi-agent system simulation of indoor scenarios. Trabajo de Fin de Grado en Ingeniería Informática (Universidad Complutense, Facultad de Informática, curso 2014/2015), 2015.

[53] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. Chapter 15: Visibility graphs. *Computational Geometry:*, pages 307–317, 2000.

[54] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[55] Dirk Helbing, Illes J Farkas, Peter Molnar, and Tamás Vicsek. Simulation of pedestrian crowds in normal and evacuation situations. *Pedestrian and evacuation dynamics*, 21(2):21–58, 2002.

[56] Ansgar Kirchner and Andreas Schadschneider. Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. *Physica A: statistical mechanics and its applications*, 312(1):260–276, 2002.

[57] Carsten Burstedde, Kai Klauck, Andreas Schadschneider, and Johannes Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3):507–525, 2001.

[58] RL Hughes. The flow of large crowds of pedestrians. *Mathematics and Computers in Simulation*, 53(4):367–370, 2000.

[59] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1160–1168. ACM, 2006.

[60] Hao Jiang, Wenbin Xu, Tianlu Mao, Chunpeng Li, Shihong Xia, and Zhaoqi Wang. Continuum crowd simulation in complex environments. *Computers & Graphics*, 34(5):537 – 544, 2010. CAD/GRAPHICS 2009 Extended papers from the 2009 Sketch-Based Interfaces and Modeling Conference Vision, Modeling & Visualization.

[61] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics*, 21(4):25–34, 1987.

[62] Craig W Reynolds. Steering behaviors for autonomous characters. In *Game developers conference*, volume 1999, pages 763–782, 1999.

[63] Abhinav Golas, Rahul Narain, Sean Curtis, and Ming C Lin. Hybrid long-range collision avoidance for crowd simulation. *IEEE transactions on visualization and computer graphics*, 20(7):1022–1034, 2014.

[64] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[65] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1928–1935. IEEE, 2008.

[66] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[67] Liang He, Jia Pan, Sahil Narang, Wenping Wang, and Dinesh Manocha. Dynamic group behaviors for interactive crowd simulation. *arXiv preprint arXiv:1602.03623*, 2016.

[68] Yiquan Song, Jianhua Gong, Yi Li, Tiejun Cui, Liqun Fang, and Wuchun Cao. Crowd evacuation simulation for bioterrorism in micro-spatial environments based on virtual geographic environments. *Safety Science*, 53:105 – 113, 2013.

[69] Stephen J Guy, Sujeong Kim, Ming C Lin, and Dinesh Manocha. Simulating heterogeneous crowd behaviors using personality trait theory. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 43–52. ACM, 2011.

[70] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 99–108, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[71] Kiran Ijaz, Shaleeza Sohail, and Sonia Hashish. A survey of latest approaches for crowd simulation and modeling using hybrid techniques. In *17th UKSIMAMSS International Conference on Modelling and Simulation*, pages 111–116, 2015.

[72] Vassilios Kountouriotis, Stelios C.A. Thomopoulos, and Yiannis Papelis. An agent-based crowd behaviour model for real time crowd behaviour simulation. *Pattern Recognition Letters*, 44:30 – 38, 2014. Pattern Recognition and Crowd Analysis.

[73] Mehdi Moussaïd, Niriaska Perozo, Simon Garnier, Dirk Helbing, and Guy Theraulaz. The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PloS one*, 5(4):e10047, 2010.

[74] MH Mabrouk and CR McInnes. Solving the potential field local minimum problem using internal agent states. *Robotics and Autonomous Systems*, 56(12):1050–1060, 2008.

[75] Legion. Science in motion. [Online; accessed Mar. 2015].

[76] Matthew Owen, Edwin R Galea, and Peter J Lawrence. The EXODUS evacuation model applied to building evacuation scenarios. *Journal of Fire Protection Engineering*, 8(2):65–84, 1996.

[77] Mott MacDonald. STEPS, 2009. [Online; accessed Mar. 2015].

[78] Thunderhead Engineering. Pathfinder, 2006. [Online; accessed Mar. 2015].

[79] Pedsim - a pedestrian crowd simulation. `http://pedsim.silmaril.org/`. (Accessed on 09/04/2017).

[80] Crowdmaster v1.3.2 - home. `http://crowdmaster.org/`. (Accessed on 09/04/2017).

[81] Epic Games. Unreal tournament 2004.

[82] Unity Game Engine. Unity - game engine. `https://unity3d.com/`. (Accessed on 08/14/2017).

[83] Edd Dumbill and Niel M Bornstein. *Mono: a developer's notebook*. " O'Reilly Media, Inc.", 2004.

[84] M Powell. JMonkey engine. *Avaliable in: http://www. jmonkeyengine. com*, 2008.

[85] T O'Brien, J van Zyl, B Fox, J Casey, J Xu, and T Locher. Maven: By example. an introduction to apache maven, 2010.

[86] Benjamin Muschko. *Gradle in action*. Manning Publications Co., 2014.

[87] C Schulte zu Berge, Artur Grunau, Hossain Mahmud, and Nassir Navab. Campvis—a game engine-inspired research framework for medical imaging and visualization. Technical report, Tech. Rep.(Technische Universität München, 2014), 2014.

[88] Zay-ES: a high-performance java-based ecs. `jmonkeyengine-contributions.github.io/zay-es`. Accessed: 2017-06-30.

[89] Michael Akroyd. Anti patterns session notes. *Object World*, 1996.

[90] Arthur J Riel. *Object-oriented design heuristics*, volume 335. Addison-Wesley Reading, 1996.

[91] Adam Martin. *Entity Systems are the future of MMOG development*. Online Article, 2007.

[92] M Mononen. Simple stupid funnel algorithm, 2010.

[93] et.al Jur van den Berg, Stephen J. Guy. snape/rvo2-java: Optimal reciprocal collision avoidance (java). `https://github.com/snape/RVO2-Java`.

[94] Autodesk Software. 3ds max | software de modelado, animación y renderización en 3d | autodesk. `https://www.autodesk.es/products/3ds-max/overview`. (Accessed on 08/10/2017).

[95] Autodesk Software. Maya | software de modelado y animación por ordenador | autodesk. `https://www.autodesk.es/products/maya/overview`. (Accessed on 08/10/2017).

[96] Autodesk. Home design and decorating ideas to get inspired and get expert tips. `https://www.homestyler.com/`. (Accessed on 08/10/2017).

[97] Blender Foundation. Free and open 3d creation software. `https://www.blender.org/`. (Accessed on 08/10/2017).

[98] Emmanuel Puybaret. Sweet home 3d, 2005. [Online; accessed Mar. 2015].

[99] Rafael Pax and Juan Pavón. Multi-agent system simulation of indoor scenarios. In *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on*, pages 1757–1763. IEEE, 2015.

[100] Luis Roalter, Andreas Moller, Stefan Diewald, and Matthias Kranz. Developing intelligent environments: A development tool chain for creation, testing and simulation of smart and intelligent environments. In *Intelligent Environments (IE), 2011 7th International Conference on*, pages 214–221. IEEE, 2011.

[101] Francisco Campuzano, Teresa Garcia-Valverde, Alberto Garcia-Sola, and Juan Botia. Flexible simulation of ubiquitous computing environments. *Ambient Intelligence-Software and Applications*, pages 189–196, 2011.

[102] Pablo Campillo-Sanchez, Jorge J Gómez-Sanz, and Juan A Botía. Phat: physical human activity tester. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 41–50. Springer, 2013.

[103] Sweet Home 3D Blog: Rafael pax interview. `sweethome3d.com/blog/2016/12/10/and_you_how_do_you_use_your_sweet_home_3d_episode_17.html`. Accessed: 2017-06-30.

[104] Kenton McHenry and Peter Bajcsy. An overview of 3d data content, file formats and viewers. *National Center for Supercomputing Applications*, 1205:22, 2008.

[105] ECMA. The JSON Data Interchange Format. techreport Standard ECMA-404 1st Edition / October 2013, ECMA, 10 2013.

[106] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM transactions on graphics (TOG)*, volume 22, pages 587–594. ACM, 2003.

[107] MakeHuman Team. Makehuman | open source tool for making 3d characters. `http://www.makehuman.org/`. (Accessed on 07/17/2017).

[108] Manuel Bastioni, Simone Re, and Shakti Misra. Ideas and methods for modeling 3d human figures: The principal algorithms used by makehuman and their implementation in a new approach to parametric modeling. In *Proceedings of the 1st Bangalore Annual Compute Conference*, COMPUTE '08, pages 10:1–10:6, New York, NY, USA, 2008. ACM.

[109] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2, 2014.

[110] Adobe. Mixamo. `https://www.mixamo.com/#/`. (Accessed on 08/04/2017).

[111] Peter M. Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. *National Institute of Standards & Technology*, 2011.

[112] Abhijit Dubey and Dilip Wagle. Delivering software as a service. *The McKinsey Quarterly*, 6(2007):2007, 2007.

[113] Andrea Giessmann and Katarina Stanoevska-Slabeva. Business models of platform as a service (paas) providers: current state and future directions. *JITTA: Journal of Information Technology Theory and Application*, 13(4):31, 2012.

[114] Sushil Bhardwaj, Leena Jain, and Sandeep Jain. Cloud computing: A study of infrastructure as a service (iaas). *International Journal of engineering and information Technology*, 2(1):60–63, 2010.

Appendix A

Papers presented

Computer Science and Information Systems 0(0):1–6 DOI: N/A

Requirement Engineering Activities in Smart Environments for Large Facilities

Jorge J. Gomez-Sanz, Rafael Pax, Millán Arroyo, and Marlon Cárdenas Bonett
jjgomez@ucm.es,rpax@ucm.es,millan@cps.ucm.es,mcardenas@ucm.es

Universidad Complutense de Madrid, email:jjgomez@ucm.es

Abstract. Developing a large, but smart environment is a complex task that requires the collaboration of experts of different disciplines. How to successfully attain such collaboration is not a trivial matter. The paper illustrates the problem with a case study where the manager of the facility intends to influence pedestrians so that they choose a task that requires certain effort, e.g. using staircases, instead of the current one that requires less effort, e.g. using the elevator. Defining requirements for such scenarios requires a strong multidisciplinary collaboration which is not currently well supported. This paper contributes with an approach to provide non-technician experts with tools so that they can provide feedback on the requirements and verify them in a systematic way.

1. Introduction

According to the International Facility Management Association [10], "Facility management is a profession that encompasses multiple disciplines to ensure functionality of the built environment by integrating people, place, process and technology". The efficiency of the use of the facility is one of the goals of a manager and for this aim, he/she must be willing to work proactively with new configurations of the facility [4]. For this aim, the incorporation of Internet of Things technology may enable a cheap enhancement of facilities that could be used to foster a more adequate use of the installation and a faster experimentation. This different use of the installation necessarily involves the alteration of the inhabitants of the facility, be them regular visitors or operators. The necessary technology and the possible, feasible, behavior alterations are questions whose answers require the collaboration of social scientists and engineers.

Basing on the hypothesis that people can modify their behaviors if the adequate stimuli are provided, a social scientist can identify a set of possible desired behaviors that could turn out. There are arguments supporting this hypothesis, as it will be discussed. Social marketing is an example of conduct alteration via visual, olfative, and auditive stimulation [15] towards the achievement of some social goal, such as improving healthy habits. However, the goal of this paper is to address the dynamic interaction with the inhabitants of a physical space.

Hence, the focus is on large groups of individuals whose mass behavior is to be subtly altered. Examples of subtle alterations may be, influencing the permanence in certain areas or preventing users to visit others. Such behavior alteration does not have to be permanent. A plan for stimulus intensity and typology is needed and means for enacting such plan provided.

Developing systems that realize this stimuli plan becomes troublesome because the gap from the requirements to the implementation is too huge. Hence, this paper illustrates ways in which this gap can be overcome using the idea of simulations as interlingua. These simulations serve to formalize the observations of the social scientists, but also to sustain the rest of the development providing a testing platform to evaluate the stimuli effect.

The use of simulations is not new in Social Sciences. However, the process by which they are created has not been studied thoroughly. Social scientists research tools are usually surveys and interviews. The way to proceed from these interviews to actual simulations is part of this contribution too. However, how these interviews and surveys translate into the system specification is not straightforward. With respect to this matter, the paper proposes using simple tools that enable to channel the effort of the social scientist into a specification workflow based on the extensive use of software simulations.

Preliminary work [11] has shown some guidelines and steps for the enactment of social sciences participation in a systems engineering process. Data and case study are reused here, but the focus is moved towards the workflow where the social scientist participates. In this sense, the original contribution are tools for retrieving feedback about the simulations and the notion of deliverable as a simulation plus comments as part of the process.

The paper is structured as follows. First, section 2 analyses if a particular behavior alteration/induction is really possible. Requirements engineering is the focus of the section 3. It proposes a set of activities to be performed within requirements elicitation, specification, and verification phases. Section 4 shows an example of how a requirements gathering is conducted in a case study of behavior modification. Section 5 introduces some of the issues found during this work. The related work is introduced in section 6. Conclusions are presented into section 7.

2. Relevant Stimulus Towards Behavior Modification

At this point, it seems necessary to address if a crowd behavior can be altered in the context of a large facility through the use of different stimuli. This argumentation is necessary step prior to engage into the analysis of the kind of behaviors that are to be enacted in a facility.

There are precedents that support such influence is possible. The first is the existence of Marketing discipline, where buyers are influenced to buy a specific item or service. This will be further discussed in the related work section (section 6).

In the conductivist research in social sciences and psychology it is possible to find evidences as well. The scope of intensity of the behavior alteration changes depending on the nature of the behavior to be modified. Subtle and small alterations, such as “staring“ gesture, have been documented. If an individual finds a group along the way, depending on its size, she will either stop, look what is happening, and stay; or keep walking [19]. The larger the group, the greater the effect. This is explained as a mirroring behavior effect. If sufficient people stare at an arbitrary point, a passerby individual will unconsciously look at the same place [8]. Gaze copying happens mainly within 2 meters range and the response depends on the physical layout of the environment, the social context, and the sex of the individual.

The impact of controlling the ”staring“ gesture of crowds in a large facility is low, specially due to the cost of having hired people to initiate the effect. However, other reactions are more relevant because of the obtained effect and the cost of producing the stimulus through artificial sources. Sound and images can affect the behavior of pedestrians. Beyer et al. [2] introduce an experiment where an interactive large banner display affects the audience. Through visual stimulus, authors manage to attract approaching pedestrians and distribute them along the display. Miller [20] shows that noise can affect people's performance. A sleepy person may be aroused by noise, but it has also negative effects, like affecting the performance of complicated tasks, affect negatively the mood and disturb relaxation. Negative effects could be used to influence pedestrians. In this paper, it is assumed that, since it can annoy people, this could be used to clear out areas or to reduce the pedestrian traffic around some places where the noise comes from.

With a more specific focus on promoting an effort taxing activity, Foster [7] reviews different works that address how the environment can promote health-enhancing physical activity (HEPA). Selected works have focused on the staircases vs elevator decision. The stimuli has been mainly posters with different kind of content and sizes and other visual aids, such as altering the aesthetics of the stairs. The effect has been to an average of a decrease of 3 points in the percentage of use of staircases. Among stimuli, videos and a combination of talking directly with individuals are not listed as used means according to [7].

Certainly, the attitude towards the stimulus depends also on the interest of the individual. The same individual may pay more or less attention to the same stimulation. A person in an airport will frequently check information panels, whereas in a mall center it may not be the case. Fun parks also influence the behavior of their visitors through information panels that tell expected waiting time for each attraction.

To understand the factors in the effectiveness of the stimulus vs the nature of the stimulus, the person, and the context, Wiebe's analysis of influencing factors in four social marketing campaigns can provide clues, as [13] quotes. There are five factors: the force (predisposition towards the goal and the intensity of the message), the direction (knowledge of how or where the person might go to consummate his motivation), the mechanism (an agency that enables to translate the person's motivation into action), the adequacy and compatibility (ability and effectiveness of the agency), the distance (the estimate of energy cost the person will invest to achieve the goal vs the reward).

Nevertheless, there are sufficient results that suggest that crowds can be influenced and its behavior modified. Humans are sensible to external stimulus. The outreach of the influence may depend on different factors, as it has been explained. In certain conditions, such as evacuations, humans pay more attention to other humans rather than other artificial elements, such as banners.

3. Requirements elicitation, specification, and verification activities

It is well known that failure in determining the requirements of a project is major cause of project failure [23]. In order to understand precisely how to achieve the kind of behavior a facility manager expects, simulations are needed and such simulations need to be validated.

To this purpose, a set of requirements elicitation actions are needed covering:

- The environment. The team has to understand the physical construction and what purposes each area serves to. Also the procedures that the facility has to pay special attention to, as the evacuation procedures. It is expected, for instance, that a cafetery gathers an important amount of traffic in a faculty building.
- Available stimuli. Literature has to be checked looking for related behaviors and associated stimuli, to evaluate the effect of each one and if there is already empirical evidence of the outcome. If the effect is not the one that permits to achieve the intended behavior alteration, a field study needs to be conducted evaluating different stimuli and their result.
- Default behavior. Documenting the initial behavior of the crowd is something very specific of social sciences. Classical methods such as recording videos, conducting surveys and interviews, can be used to gather information. A later analysis is needed anyway to evaluate the information and find clues of what stimuli could be more effective towards achieving the desired behavior.

This information captured at this stage is computationally represented through simulations by an engineer. This begins the requirements specification phase. However, the simulations have to deal with incomplete data. Tracking the movement of the individuals moving through a facility is not possible in general, due to budget or legal constraints[22]. Usually, the information about the movement of the people inside a building is given in form of datasets of pedestrian traffic measured on strategic checkpoints over time. Hence, it is a major issue how to imply or infer what behavior the untracked people people had along the experiment. To address this issue, it is assumed that more than one simulation have to be produced, each one identifying pedestrian behaviors traits as precisely as possible. Following sections will introduce how an algorithm for population behavior is used to generate such initial simulation setups. These setups suggest different trajectories that satisfies the empirical data collected in the field experiment or available literature. Whether these simulations are realistic, in terms of expectations of human behaviors, is something that is evaluated later on by an expert.

If these computer simulations captured the reaction of the crowd when the relevant stimuli was produced, then they could be used as testbed for evaluating what stimuli sequence would be needed in each situation. To achieve this, basic reactions of the simulation actors need to be accurately represented so that, during the simulation and while producing the stimuli, the reaction of the crowd is a convincing one. To this aim, some basic reactive behaviors are coded in the simulated characters. Section 4.2 discusses this aspect.

The interplay between the engineers and experts is introduced in figure 1. Engineers produce the simulation that captures the expected behavior of the crowd,while the experts validate the simulations.

The simulation specification is performed by an engineer. The engineer will define a set of possible actor behaviors, an enumeration of the number of instances of these behaviors, and a timestamp of when the behavior & actor instantiation happens.This permits to define scenarios where simulated actors arrive to the facility at different times. Simulated actors will either be responsible of operating the facilities or just visitors. The first group will be in charge of coordinating the behavior of the second group, just as a security guard guides people outside the building in an evacuation, for instance. Even though there maybe a role switch may happen in the literature, e.g. a citizen becomes a leader

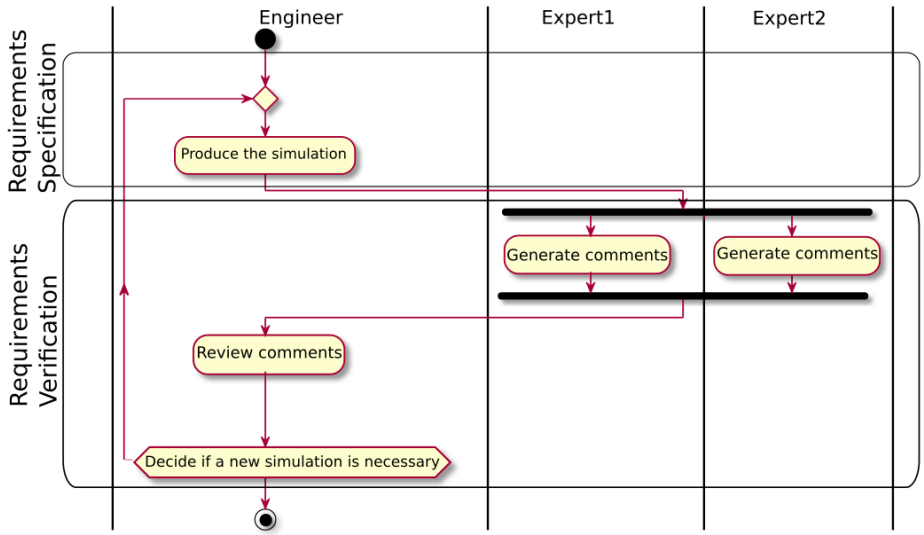

Fig. 1. Activity diagram showing specification and verification activities

in an evacuation, it is not considered in this paper. Simulated actors conduct a specific predetermined behavior sequence, due to scalability issues.

Each possible predetermined behavior represents a role of a person in the facility (e.g visitors, people working on the facility or security staff). They are called *predefined behaviors* because they define roughly the behavior of the role. Every actor behaving as a ”student” will behave in the same way as other students. However, future versions may consider small behavior variations.

As part of the requirements verification activities, it will be necessary to ensure the consistency and validity of the simulation.It is likely that many different simulations are produced as consequence of grounding variables such as the initial population or daily activities. The verification of all produced simulations can be achieved with the collaboration of the social scientists that produced the initial data. They need to observe the result of the computer simulation and subjectively decide if the simulation makes sense or not. According to figure 1, these experts are expected to produce approving/disapproving about the simulation.

Having an expert in sociology or social science can save the technician time and ensure an assessment of the model more in line with human behavior theories. For example, Boukas, Evangelos et al. [3] propose the same concept of social distance proposed in this paper, based on the field of proxemics, for an analog model of pedestrian traffic simulation. The collaboration of experts for the verification and validation of pedestrian traffic simulations is considered too in [6]. They suggest that the evaluation of the models requires a large amount of detailed data whose production consumes time and effort, making the identification and design of suitable criteria and data especially important. The involvement of these experts is not trivial either, as indicated in [5]. They provide an easy-to- use simulation environment for non-computer users for validation and verifica-

tion. This type of environment that is easy to understand and use for any user would be the one that would give greater facilities so that the specialists of other fields outside the computational sciences to express and simulate to carry out the work of verification and validation of the model, as well as to propose improvements.

In any case, literature supports the hypothesis that experts are necessary, that they need to be involved to select the most relevant criteria and data, and that they need appropriate tools to do so. Though social scientists have been cited so far, other professionals could be added too. Psychologists as well as experts in safety and security can be included in the gathering and elicitation of requirements.

They need to observe the result of the computer simulation and subjectively decide if the simulation makes sense or not. Though social scientists have been cited so far, other professionals could be added too. Psychologists as well as experts in safety and security can be included in the gathering and elicitation of requirements.

4. The case study

As a case study of facility management support scenario, two university faculties are considered: a computer science faculty and a social and political sciences faculty. In each faculty, it is intended to change the habit of using the elevator for a more healthy one which involves using more the staircases. Since the experiment deals with two different populations, it is an opportunity to study the reaction of two groups of people when facing the same stimuli. These two groups have relative homogeneity within each faculty, but there differences when comparing one to each other.

The homogeneity of the two groups is given by the characteristics of its members, who are students, teachers and administrative staff of the same university. However, there are differences that need to measured: the daily activity of the faculties is different (e.g. schedules) and the attitude differs too when considering specific groups, such as students. The schedule, for instance, is more intensive in social and political sciences faculty from Monday to Thursday. Also, the building hosts an additional faculty, what increases the diversity in the schedules and the occupation of classrooms. The attitude differs too in the daily activities when considering students. As an example, in the social and political sciences faculty students more frequently express themselves in public through statements and public meetings. However, in the computer science faculty, the behavior of students is less reinvindicative. Besides, the student associations do involve themselves mainly into organizing computer related activities into closed rooms, rather than meeting into corridors or halls. The field study would point out quantitatively how these observed differences do affect the daily activities within these buildings.

An important part of the field study is the measurement of the effect of the stimuli in the crowd. Variables such as the already mentioned differences between the populations in the two buildings ought to affect the effectiveness of the stimuli. Through a phased field experiment, the effect of each stimuli had to be measured, so that it could be later on reproduced.

4.1. Requirements elicitation

Starting with the requirements elicitation activities stated in section 3, the environments were analyzed. In this case, both faculties were inspected looking for locations for running

the behavior altering experiment. In this case, it required studying the behavior patterns of students and staff, looking for the elevators that were more used apparently, and that had staircases close by. For each location, observation points at different floors were identified too. Also, experimentation days were chosen taking into account when students and staff could have a peak in the occupation of the building.

The available stimuli in the literature was studied too. Some results have been cited here already in section 2 and others will be included in the related work section. Mostly, literature referred to banners situated along the path of pedestrians. For the experiment, three different stimuli are identified: banners containing some information that may affect the pedestrians; multimedia beamers showing videos motivating to show the staircase; and direct intervention. Different **banners** prototypes were made. Following the literature, it was intended to highlight facts that may be relevant to pedestrians, see figure 2. In summary, the banner suggests less time to reach the destination, more climate friendly activity, and health benefits because of the calorie burning. For the **multimedia**, a vertical beamer was used. This beamer projected over a panel with sufficient surface to attract the attention. It was the equivalent of a 55" screen. It played a video made for this occasion. It was a dramatization of a person that can walk, but does not want to, even wants to use a regular chair inside an elevator, and asks for people to raise the chair and get the person out. The **direct intervention** was made by volunteers that acted as if they were a *facility operator* trying to influence a visitor. They stopped pedestrians with the excuse of doing a survey, but, in fact, they asked the users if they knew of the benefits of using staircases.

Fig. 2. Banner stimulus with information concerning the benefits of using staircases. It is written in Spanish. The main title says *stair climbing and avoiding elevators* at the top. The alleged reason are *1. improving your health*, *2. You will get faster to your destination*, and *3. you will save energy*

To evaluate the effect of the stimulus over the two faculties, a five week schedule was prepared:

- First week (week A). There was no stimulus and it was used to collect a base line of staircase/elevator traffic stats
- Second week (week B. The banner stimulus was introduced
- third week (week C1). The videos were added to the experiment
- Fourth week (week C2). Volunteers stop pedestrians to ask them if they know the benefits of going upstairs. To prevent rejection, the question is disguised as a survey.
- When week C2 finishes, all stimuli are removed from the environment.
- Fifth week (week D). This week happens a few days after week C2 finishes. The goal is to measure how much the behavior persist without stimuli reinforcement.

Along the experiment, a team of observers recorded in specific locations of the buildings how many people were traversing per minute the area and if they were going to get on or off the elevator, and if they were going upstairs or downstairs. By measuring the people crossing these sections, the effect of the stimuli could be determined. Measurements and stimulation were made the same days each week, to replicate the same conditions as much as possible every time. The accumulated data, obtained from [11], is presented in table 1.

The number of people arriving through the elevator remains mostly the same along stages. However, the number of people choosing not to use the elevator when going upwards is reduced from 29.3 to 25.3 points in the percentage in phase C2. This variation is consistent with other experiments found in [7], where similar variations were found. In relative terms, the variation of 13,65% over the original use of the elevator. To value the results, it should be taken into account that each stimulus lasted for one week, and not months.

Table 1. Variation of the traffic in elevators in two faculties as introduced in [11]

% use elevators	A	B	C1	C2	D
Total	23,1	21,9	21,4	20,3	22,2
Departures	29,3	28,2	26,2	25,3	26,4
Arrivals	14,4	14,4	15,2	14,0	16,2
#total=	9730	9797	9459	9165	9088
#departures=	5688	5371	5335	5109	5345
#arrivals=	4042	4426	4124	4056	3743

With the end of the field experiment and the analysis of the obtained data, the requirement elicitation finishes.

4.2. Requirements specification

The activities in requirements specification and requirements verification are expressed in figure 1. The subject of this section are the requirements specification part and will be repeated as the verification indicates the simulation or simulations are not correct.

As explained in section 3, the behavioral data collected by social scientists have to be translated into simulations that reproduce the observed behavior. The whole population movement is unknown and if only the information gathered during the field study is used, there is a potential unlimited number of simulations whose behavior matches the data. For

instance, if the observed data tells a section was crossed twice, a simulation that would fit the data could be one where there is a single individual crossing twice that section; but another possible simulation would have two individuals crossing once the same section. Hence, the simulation specification poses a strong parameterization problem.

Actors in the simulation are playing the visitor role and there are none with *facilities operator*. Their basic behavior consists of entering the building, visiting some rooms, and then getting out of the building. During parameterization, a number of visitors is chosen, and for each visitor, the number of rooms to be visited and the sequence, and how much time will be invested in each location.

Actors move across the terrain facing collisions with obstacles and among them. The specific path, speed, and delays that the actor finds adds some uncertainty to the result. There is no coordination in the current version of the simulation, so actors do not agree on the path. However, there is some implicit agreement when considering imminent collisions among two individuals. There is a path replanning to avoid the collision, which is close to what humans do.

Under these constraints, a first goal is, given a predetermined population size, find what behavior has to be assigned to each individual so that the execution of the resulting navigation path provides the same traffic data as observed. Once this goal is achieved, the next step is to reproduce the stimulus effect on the individual actors to obtain variations in the observed traffic similar to the experimentation, i.e., a 4 percentage points of variation of the traffic at the observed locations.

At the end of the process, the resulting traffic in all stimulus scenarios ought to match those of table 1. Achieving this turns out to be challenging because of the unexpected collisions, accumulation of individuals in crowded places, or the integration of elevators in the problem. About the later ones, elevators, see figure 3, are one of the most difficult problems to model. They require the simulated actors to coordinate the arrival and the evacuation of the elevator. If the elevator stops at the first floor, the simulated actor wants to get to the second, but the actor is at the front part, it should either get out or move aside to let others exit first.

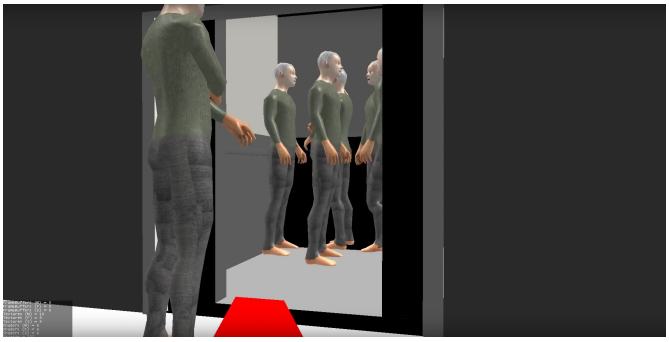

Fig. 3. A capture of the incoming and outgoing traffic to the elevator

Even if there is a simulation that matches the observed field study data, from the perspective of the social scientist, it may not make sense. As an example, the movement of characters may look unnatural if all characters decide to turn around in the middle of a corridor. That is the reason why, even though a simulation is data-wise correct, it may not be the one the project pursues. This problem has been studied in [21] and an algorithm proposed that generates a simple population.

To observe the simulation, several cameras are needed. Figure 4 shows an evolution of the work done in [11]. On the left hand side, there are five selectable cameras that show different locations of a building. On the main screen (center), the principal traffic location is shown, which matches the main elevators and staircases. The building matches the faculty of computer science in this case. The design of the simulation is such that the cameras, in this case, are pointed out to the places where observation data was collected. The bottom part of the interface shows the recorded data against the real time generated traffic data. Ideally, both lines (generated and recorded data) ought to be close, but there is a variation. The reason is the above mentioned unexpected collisions and bottlenecks.

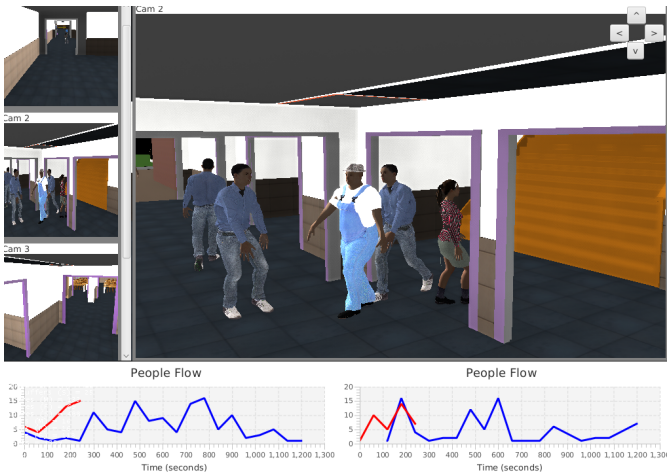

Fig. 4. User interface of the simulator, showing different cameras that permit to observe the behavior of people and charts to summarize quantitative data on some events. There are two lines, a complete line in blue representing the measured pedestrian traffic in the real world, and another growing line in red representing the measured traffic along the simulation.

Stimuli and simulations An important problem consists in quantifying the reaction of the people to the different stimuli, as measured by the field study. The simulations, so far, reproduce separately specific phases of the experiment, phases A to C2 according to table 1. Hence, the developer has to produce as many simulations as phases considered along the experiment. In each simulation, the goal is to reproduce the empirical data through a simulation.

A later step consists in experimenting with the simulations alone, altering a simulation corresponding to one field experiment phase to include stimuli considered into another phase. This is a more complex step not considered in this paper.

The course of action would involve including reactive behaviors into each simulated actor. Such individual reaction is already coded in the current version. For instance, this enables a simulated character to avoid collisions with other characters, or to alter the navigation when some unexpected bottleneck arises in a corridor.

4.3. Requirements verification

The activities in requirements verification included in figure 1 deal with the analysis of the simulations to determine whether they are correct or not.

The obtained simulations, as those from figure 4, need to be evaluated by the experts, social scientists in this case. For this goal, a way for delivering the simulations to the experts and get feedback was necessary. Figure 5 introduces the tool. The simulation is re-run to obtain a separated video per camera. These videos are integrated in an HTML5 + JavaScript application. The reason for an HTML5 + JavaScript implementation has the advantage of working in almost any computer as long as there is a browser.

The expert has to review the different videos which are played in a synchronized way. The review is aided with controls for freely moving across the videos: a slider control for jumping to specific sections; play and pause buttons; a framerate modification button; and controls for stepping forwards/backwards a few frames.

The expert has to visually inspect the simulation and then annotate in each captured camera whatever information to be recorded as feedback. Annotations are associated to the current timestamp and are specific of a camera feed. The expert can select an annotation and make the simulation move to the associated timestamp, what implies moving forwards or backwards all videos. To facilitate the later review of the comments, the expert can add a flag to the annotation to express whether the annotation is positive (the simulation is correct at that point) or negative (the simulation is incorrect). The positive annotation is not required every time and it is intended for informing that a failure in a previously reviewed simulation has been fixed. They mean that the problem has been solved. Also, they are useful to point at behaviors that have to be preserved till the end of the development of the simulation, like a regression test.

Once the video is annotated, the expert can store the changes and send them back to the team. Then, the annotations can be checked looking for negative annotations. These are reviewed and included in the documentation of the development as a *deliverable*.

For example: In the context of a simulation about daily activities in a building, the behavior of the simulated users, at certain simulation time, might not match the expectations of the social sciences expert, such as having people insisting in getting closer to a crowded exit. The social science expert would address this issue through the annotation tool described previously. This expert would add a negative comment pointing out the irrationality of such behavior. Later on, a review of the comments of the expert would aid the development team to identify which individual or group behaviors are incorrect. In this case, the solution is to make simulated users avoid getting stuck into crowded exits. Then, the development team would produce a new simulation addressing these issues. The expert would review again and convert the negative comment into a positive one. Along the process, the produced artifacts would be an initial simulation, some comments

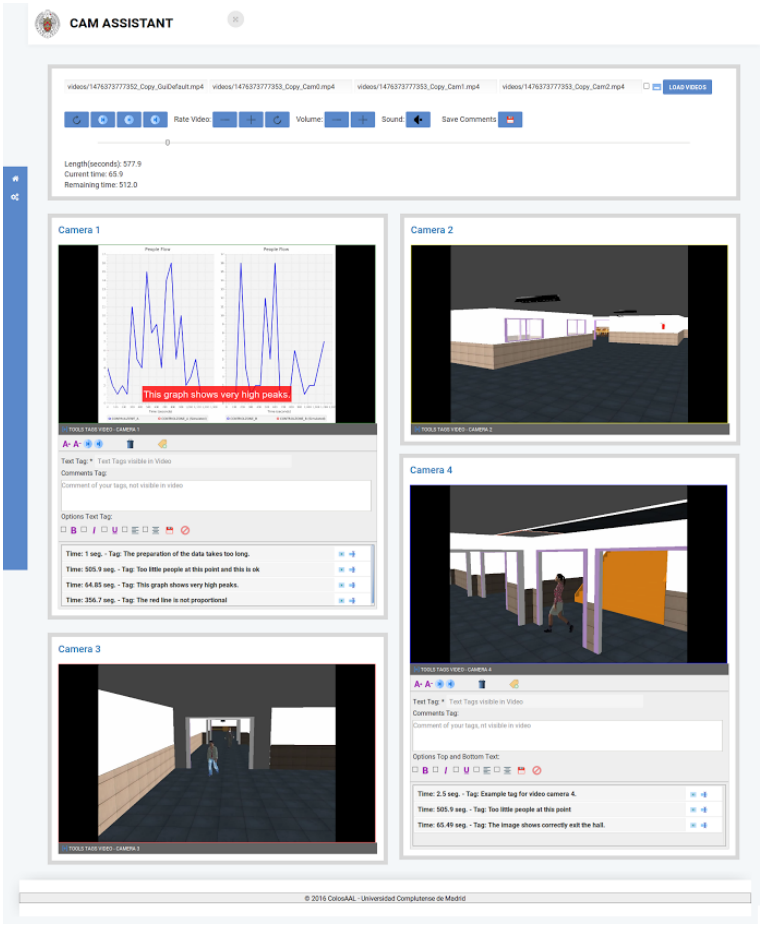

Fig. 5. Annotation tool for adding comments to the simulation

(positive and negative) generated by an expert, and a new simulation that addresses the comments.

This manual validation made by experts is necessary because the simulation may contain inconsistent results. Human experts can notice these inconsistencies and decide if they really happen in the real world or not. As an example, a simulation may have most simulated actors gathered at the lower floors and having little or no people at the upper ones [21]. In the real building, top floors do not have classrooms, only offices. If top floors really were empty, the facility manager may consider if the building was being effectively used or not. However, the human expert may think too that the simulation is wrong and that the building cannot have empty top floors. In this case, top floors are almost empty and have less pedestrian traffic than lower floors.

5. Evaluation

Generating an annotated version of the simulation and defining the simulation itself is proving to be an effort taxing task, mainly because of the inherent technological problems when defining the behavior of the crowd. It is not the definition of the population, but the management of the unexpected events such as collisions and bottlenecks what is adding cost to the development.

The work done has not taken into account stakeholders, such as the *facility manager* or the staff in the facility. However, it is expected that the simplicity of the process in annotating videos makes the task affordable in terms of human-to-computer interaction. The feedback mechanism is the same despite the background of the expert. Only the content of the annotations will differ, which is something expected due to the different perspectives of each reviewer.

The availability of different cameras makes the review simpler. In the GUI, see figure 4, the user can switch cameras by selecting one on the left. The annotation tool, see figure 5, is more flexible, since the user can reorganize the videos and even make the window larger to see them all at once. However, it will require a larger monitor.

Processing these annotations is another time consuming task. It is not only about reading the comments, but also classifying them and making sure that new simulations do not fall into the mistakes while still maintaining the behavior the positive annotations identified. By now, this needs to be done manually, but it has pointed out a non trivial problem, which is how to formally capture the comments so that they can be automatically verified across simulations.

About the behavior of the simulated crowds and the field experiment data from figure 4, there is still not a perfect match. Improvements made to work in [11], such as having different people sizes (people in the figure 4 is shorter than people in figure 3) has created greater uncertainty in the outcome of the simulation since the collisions are more frequent. This implies simulated characters take longer to achieve the destination, so they cross the control sections at a different time. This naturally affects the final accounting of simulated pedestrian traffic. This problem will become harder after adding the social concerns that the simulation must include, such as constraining the movement of individuals to fit into social-etiquette conventions. For instance, if the place is not crowded, two persons will be close if they know already each other, or one want to address the other. Otherwise, both will keep the distance. Other relevant concerns are moving in groups or pairs, avoiding bottlenecks, or dealing with the exit of rooms in an ordered way.

6. Related work

There are works dealing with the design of smart systems, but they do not frequently consider human sciences and stimulus to plan the kind of system which is needed and what performance it will have. Harrison [9] claims the analysis of mutual and incidental user interaction has not been accounted and proceeds to apply fluid flow analysis to understand it. This kind of analysis is necessary, but, it does not replace a more conventional study and cannot assume a 100% response of the individuals every time. Other works focus on the devices expected to provide the stimulus at small scale, such as [24]. Though authors

stress the involvement of human scientists too, the behavior of people in small spaces cannot be compared to that of large spaces.

There are precedents too in reproducing observed data as simulations. In [14], video recordings were used to reproduce later on a crowd simulation of simulated actors. Behavior of the individuals were obtained from a multiple checkpoint observation that allowed to reproduce the pedestrian traffic of the facilities where the measurements were made.

The project introduced in this paper, however,assumes incomplete information about activities and traffic. The less information is used, the less expensive a real installation would be. Following the same paradigm, Lerner et.al [16] propose the creation of an example database for evaluating simulated crowds based on videos of real crowds. Bera. et.al [1] also developed a behavior-learning algorithm for data-driven crowd simulation, capable of learn from mixed videos. Zong et.al [25] developed a framework for generating crowds for matching the patterns observed on video data,taking into consideration the behavior both at the microscopic level as at the macroscopic level. Finally, Yi Li et.al [17] developed a technique for populating large environments with virtual characters, cloning the trajectories of extracted crowd motion of real data sets to a large number of entities.

One discipline that has extensively researched the behavior of people while wandering through large commercial facilities is Marketing. Marketing is about the exchange process, "where two or more parties, each having something to exchange, and both able to carry out communications and distribution" [13]. Then, marketing management could be defined as "the analysis, planning, implementation, and control of programs designed to bring about desired exchanges with target audiences and the purpose of personal or mutual interest" [13]. In a way, Marketing wants to modify a behavior, a buying behavior, of a target group of people, the target customers. The approach could be applied to other less business-like goals, such as increasing charity donations or teaching cultural institutions how to attract new sponsors. It is the social marketing [13][12], and pursues "the design, implementation and control of programs calculated to influence the acceptability of social ideas". Though this goal differs from the project, there are points in common in the use of stimulus to alter the behaviors. More recently, McKenzie-Mohr [18] uses marketing to raise concern on climate change and induce less contaminating behaviors. Marketing provides valuable lessons in how to analyze and handle the stimuli, though from the perspective of business orientation, classic Marketing, or a short term controlled influence, like social marketing. The work introduced in this paper is more related to long term stimuli production and a dynamic configuration of the stimuli to adequate the behavior of the crowd according to the requirements expressed by stakeholders such as a *facility manager*.

7. Conclusions

The paper has worked on specific phases of Requirements Engineering to clarify the role of social scientists in the development and introduced tools that help to achieve the goal of formally documenting with simulations a crowd behavior alteration scenario.

Previous work [11] was oriented towards incorporating simulations into a larger development guideline. This paper has contributed with a special focus into requirements engineering activities, in particular requirements specification and requirements validation.

For this aim, a set of activities have been suggested that depend on a specific annotation tool introduced in this paper.

The collaboration of experts, in this paper social scientists have mentioned, is a critical element. Since the project aims to capture the reaction of the crowd towards stimuli, an expert needs to assess that the simulation is actually showing a similar reaction. Also, the expert has to approve that a simulation is actually representing a real behavior. In this work this is even more important, since the simulation is built using partial pedestrian traffic data. Experts need to evaluate whether the recreation produced by the engineer actually resembles a real behavior.

Correctly capturing this feedback provided by the collaborating experts is an important step in order to guarantee the quality of the final development. Since the requirements engineering activities do extensively use simulations as formal specification tool, this paper has proposed to understand this feedback process in form of annotations made to a video that represents the simulation of a particular scenario. There are several advantages for this approach: the problem is reduced to something that requires no additional means aside a browser; the actions required on behalf the expert limit to annotating a video, something that should be intuitive enough; and the involved artifacts, videos and comments, can be easily stored and reproduced anytime, also they can be subject of configuration management activities, such as controlling their evolution through version control tools.

However, the generation of simulations itself remains a specialized activity that still requires the participation of engineers. It remains as future work to develop means that facilitate the online creation of these simulations and, perhaps, the collaboration among experts to create them.

In the paper, simulations reproduce empirical pedestrian traffic of specific field experiment phases. Hence, there is at least one simulation per field experiment phase. It remains as future work to generate simulations where an operator can tune the behavior in runtime, add the stimuli to the simulation, and observe how the crowd behavior changes to fit experimental data.

Another challenge is how to systematically deal with sets of annotations produced by different experts. A development team may not have the criteria to decide, when conflicting comments arise, which one should be the more chosen in the new simulation. It is expected that techniques like focus group and other qualitative analysis techniques help to organize and prioritize the different comments.

8. Acknowledgements

We acknowledge support from the project “Collaborative Ambient Assisted Living Design (ColoSAAL)” (TIN2014-57028-R) funded by Spanish Ministry for Economy and Competitiveness; and MOSI-AGIL-CM (S2013/ICE-3019) co-funded by Madrid Government, EU Structural Funds FSE, and FEDER.

References

1. Aniket Bera, Sujeong Kim, and Dinesh Manocha, ‘Efficient trajectory extraction and parameter learning for data-driven crowd simulation’, in *Proceedings of the 41st Graphics Interface Conference*, pp. 65–72. Canadian Information Processing Society, (2015).

2. Gilbert Beyer, Vincent Binder, Nina Jäger, and Andreas Butz, 'The puppeteer display: attracting and actively shaping the audience with an interactive public banner display', in *Proceedings of the 2014 conference on Designing interactive systems*, pp. 935–944. ACM, (2014).
3. Evangelos Boukas, Luca Crociani, Sara Manzoni, Giuseppe Vizzari, Antonios Gasteratos, and Georgios Ch Sirakoulis, 'An intelligent tool for the automated evaluation of pedestrian simulation', in *Hellenic Conference on Artificial Intelligence*, pp. 136–149. Springer, (2014).
4. Kathy O ROPER CFM, Richard P PAYANT CFM, et al., *The facility management handbook*, AMACOM Div American Mgmt Assn, 2009.
5. Mizar Luca Federici, Lorenza Manenti, and Sara Manzoni, 'A checklist for the evaluation of pedestrian simulation software functionalities', *arXiv preprint arXiv:1404.7717*, (2014).
6. Mamy Fetiarison, Gunnar Flötteröd, and Michel Bierlaire, 'Evaluation of pedestrian data collection methods within a simulation framework', in *AET, European Transport Conference. Glasgow, Scotland. 11/10/2010–13/10/2010*, (2010).
7. Charles Foster and Melvyn Hillsdon, 'Changing the environment to promote health-enhancing physical activity', *Journal of sports sciences*, **22**(8), 755–769, (2004).
8. Andrew C Gallup, Joseph J Hale, David JT Sumpter, Simon Garnier, Alex Kacelnik, John R Krebs, and Iain D Couzin, 'Visual attention and the acquisition of information in human crowds', *Proceedings of the National Academy of Sciences*, **109**(19), 7245–7250, (2012).
9. Michael D Harrison, Mieke Massink, and Diego Latella, 'Engineering crowd interaction within smart environments', in *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pp. 117–122. ACM, (2009).
10. International Facility Management Association. https://www.ifma.org/about/what-is-facility-management, October 2016.
11. Rafael Pax Jorge J. Gomez-Sanz and Millán Arroyo, 'Towards the simulation of large environments', in *1st International Workshop on Ambient Intelligence for Large Premises (AmILP 2016)*. CEUR Proceedings.
12. Philip Kotler and Eduardo L Roberto, 'Social marketing. strategies for changing public behavior.', (1989).
13. Philip Kotler and Gerald Zaltman, 'Social marketing: an approach to planned social change', *The Journal of Marketing*, 3–12, (1971).
14. Kang Hoon Lee, Myung Geol Choi, Qyoun Hong, and Jehee Lee, 'Group behavior from video: A data-driven approach to crowd simulation', in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pp. 109–118, Aire-la-Ville, Switzerland, Switzerland, (2007). Eurographics Association.
15. Nancy R Lee and Philip Kotler, *Social marketing: Influencing behaviors for good*, Sage, 2011.
16. Alon Lerner, Yiorgos Chrysanthou, Ariel Shamir, and Daniel Cohen-Or, 'Data driven evaluation of crowds', in *Motion in Games*, 75–83, Springer, (2009).
17. Yi Li, Marc Christie, Orianne Siret, Richard Kulpa, and Julien Pettré, 'Cloning crowd motions', in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 201–210. Eurographics Association, (2012).
18. Doug McKenzie-Mohr, *Fostering sustainable behavior: An introduction to community-based social marketing*, New society publishers, 2013.
19. Stanley Milgram, Leonard Bickman, and Lawrence Berkowitz, 'Note on the drawing power of crowds of different size.', *Journal of personality and social psychology*, **13**(2), 79, (1969).
20. James D Miller, 'Effects of noise on people', *The Journal of the Acoustical Society of America*, **56**(3), 729–764, (1974).
21. Rafael Pax and Jorge J. Gómez-Sanz, 'A greedy algorithm for reproducing crowds', in *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection, 14th International Conference, PAAMS 2016, Sevilla, Spain, June 1-3, 2016, Special Sessions.*, pp. 287–296, (2016).

22. Qasim Mahmood Rajpoot and Christian Damsgaard Jensen, 'Video surveillance: Privacy issues and legal compliance', *Promoting Social Change and Democracy Through Information Technology*, 69, (2015).
23. Roy Schmidt, Kalle Lyytinen, and Paul Cule Mark Keil, 'Identifying software project risks: An international delphi study', *Journal of management information systems*, **17**(4), 5–36, (2001).
24. Norbert A Streitz, Carsten Röcker, Thorsten Prante, Daniel Van Alphen, Richard Stenzel, and Carsten Magerkurth, 'Designing smart artifacts for smart environments', *Computer*, **38**(3), 41–49, (2005).
25. Jinghui Zhong, Wentong Cai, Linbo Luo, and Haiyan Yin, 'Learning behavior patterns from video: A data-driven framework for agent-based crowd modeling', in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 801–809. International Foundation for Autonomous Agents and Multiagent Systems, (2015).

Agent-based Simulation of Crowds in Indoor Scenarios

Rafael Pax and Juan Pavón

Abstract Crowd simulation models usually focus on performance issues related with the management of very large numbers of agents. This work presents an agent-based architecture where both performance and flexibility in the behaviour of the entities are sought. Some algorithms are applied for the management of the crowd of agents in order to cope with the performance in the processing of their movements and their representation, but at the same time some alternative reasoning mechanisms are provided in order to allow rich behaviours. This facilitates the specification of different types of agents, which represent the people, sensors and actuators. This is illustrated with a case study of the evacuation of the building of the Faculty of Computer Science, where different types of human behaviours are modelled for these situations. The result is the simulation of more realistic scenarios.

1 Introduction

There are several models for crowd simulation but in general these focus on scalability issues derived from the management of a large number of agents in real time, specially when considering their visualization or the way the agents find their way while avoiding obstacles and other agents [1]. Different techniques have been proposed to cope with these issues, by relying on specific assumptions of the problem under study. This has, however, an effect on limiting of the flexibility of agents' behaviour, which is quite homogeneous in most of the cases.

In this work the scope of the problem is the simulation in indoor scenarios, where the number of agents may be large (thousands) but not very large as in a city (millions). This gives more room on performance constraints, which opens the possibility to modelling of individual agents with heterogeneous behaviours.

Universidad Computense Madrid (Spain)
e-mail: {rpax, jpavon}@ucm.es
http://grasia.fdi.ucm.es

Several tools for simulation and design of how people behave in indoor scenarios exist [9, 14, 15, 16, 18], but they are still considering the agents more like a crowd that can be characterized by simple behaviours with a fixed number of parameters, instead of considering them as individuals. Other works, like [10, 8, 11, 12, 13] have addressed the specification of richer agent behaviours, but the methodological aspects for a design process when developing them are not sufficiently exposed.

Taking this into account, this work proposes an agent-based model for indoor scenarios where both performance and flexibility in the behaviour of the entities are sought. Agents are specified and managed individually, but the effects of the crowd are taken into account by several methods that take advantage of characteristics of the indoor domain in order to cope with the efficiency and scalability issues in the processing of their movements and their visualization. At the same time, some alternative reasoning mechanisms are provided for each agent in order to allow modelling of rich and heterogeneous behaviours.

Section 2 introduces the MASSIS (Multi-agent System Simulation of InDoor Scenarios) architecture and components. They allow for the specification of the elements for indoor scenarios simulation. Special attention is given to the definition of the behaviour of humans under different situations, which includes the process for decision making of the agents. Other relevant aspects to model are interactions among agents and with their environment, the events on the environment, and the precise representation of the building.

The strategies that facilitate an efficient management of crowd simulation aspects are presented in Section 3. This is illustrated with a case study on the evacuation of the building of the Faculty of Computer Science in Section 4. The model facilitates the specification of different types of agents, which representing the people, and the sensors and actuators of the environment. Different views can be generated and manipulated during the simulation. Finally, Section 5 presents the conclusions.

2 Overview of MASSIS architecture

MASSIS has a component-based architecture, where some part of the infrastructure is well-proven open source software. An agent-based framework above of the core infrastructure allows the specification of flexible agent behaviour types, as well as interactions among agents and the elements of the environment.

The environment initially is defined using SweetHome3D, a well known package that is used to model all components involved in an indoor environment, such as walls, doors, stairs, people, etc. In order to facilitate more flexibility of the characterization of the physical elements, it is possible to define a set of plugins, to specify the characteristics of the elements of the building, which will be represented by agents. For instance, in the case of sensors and actuators, they will be reactive agents, with simple behaviour and attributes. In the case of people, some physical characteristics can be defined such as weight, speed, but also some inherent attributes of the per-

son (fear, courage, etc.) and a link to their behaviour. The types of behaviours are defined as components.

The simulation engine of MASSIS is MASON [5], a lightweight multi-purpose agent-based simulation library. Agents' behaviour is controlled with the Pogamut's POSH engine [4]. Some tests have been performed in order to check the performance of their integration in MASSIS. In the order of 10 thousand agents the experimentation has shown that execution times grow linearly with respect to the number of agents, so the results are satisfactory.

All the changes made in the environment are reflected in real time by 3D (Fig. 5) and 2D (Fig. 2, 3, 8) displays, and can be logged in JSON format, as a single zipped file or in a SQLite database for further analysis. Although 3D display is more realistic and nicer for demonstration purposes, the 2D view is useful for analysis and debugging. Also, the 2D visualization API allows the creation of user-defined layers in order to filter the different elements involved in the simulation.

Once a simulation is performed, the exported data can be used to playback all events that have occurred during the execution of the simulation, i.e., the agents will behave in the same way they did during the simulation. This is interesting to allow the users to review the simulation when analysing what has happened.

Modelling human behaviour with agents have to consider two main aspects: those related with their perception and the interaction with the environment, and those dealing with the reasoning on the context and the decision making. They are implemented as low-level and high-level behaviour components, respectively. When the high level component decides *what to do next*, the action is executed by the low-level component, which performs all the necessary operations. The relationship between these components is illustrated in Fig. 1

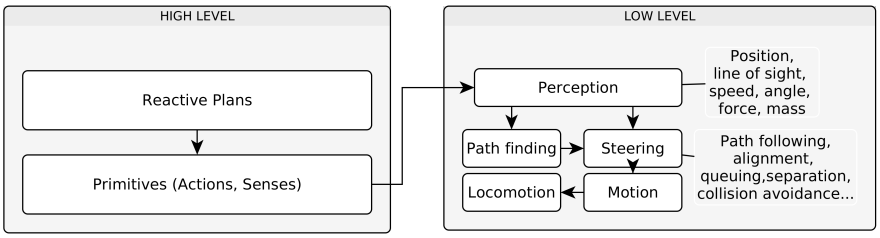

Fig. 1 MASSIS 's human behaviour agent model.

The low-level components deal with the perception of the environment and a set of basic behaviours for interacting with it. These behaviours are mostly a combination of *steering behaviours* [2], which are explained in Section 3.

The high-level behaviour components deal with decision making, learning and communications with other agents. Decisions are taken on the knowledge of the environment, which is provided by the low-level components.

Behaviour Oriented Design (BOD) [3] is applied to facilitate the design of agents. Developing a system of agents under BOD involves dividing their implementation

into two different parts: A library of Behaviour modules and POSH Dynamic action selection scripts. Behaviour modules are a set of classes representing a set of modules for perception, action and learning (i.e *primitives*).They can be called from the mechanism of action selection in order to determine *how* to do something. These senses and actions are created in the native language for the problem space (in the case of MASSIS, Java).

A POSH action selection script is a prioritized set of conditions and the related actions to be performed when certain conditions are satisfied. Figure 6 shows an example with the main elements: *drive collections*, *competences*, and *action patterns*. On the action selection step, the POSH engine executes the corresponding action pattern or competence from the drive collection with highest priority. Competences are similar to drive collections, but rules they do not interrupt each other. Finally, action patterns are simple, reusable sequences of actions.

The next sections present some of the components of the MASSIS framework that facilitate the efficient management of crowds of agents in a simulation.

3 Crowd Modelling and Simulation Issues

There are several aspects to take into account when modelling crowds of agents, with a trade-off between efficiency and flexibility of the specification of behaviours.

The building model, designed with SweetHome3D, is loaded and transformed into MASSIS internal representation to support the efficiency of algorithms implementation. This has an impact in the way agents interact with the environment, which is described below for different aspects: path finding, localization of elements and steering behaviours.

3.1 Path finding

Pathfinding is one of the issues that has more impact when simulating crowd behaviours. Some models treat the crowd as a single entity or a group in order to simplify the number of calculations, such as in [6]. However, MASSIS, as it has been stated in the introduction, has as objective to support flexibility in agent behaviour, therefore the path finding model is implemented individually for each agent, but taking advantages of some assumptions from the problem domain in order to gain in efficiency.

When the action selection component (see Figure 1) decides that the agent must go to a particular location, a path must be computed from the agent's location to the target. Its computation is done in MASSIS by an A* search over the polygons' visibility graph. The computational cost has been considerably improved by taking advantage of several characteristics of the environment:

1. The perimeter of rooms consists of walls or doors.

2. The path from a point A to a point B, being B in a different room than A, must pass necessarily through a door.
3. In an indoor scenario walls intersect each other *very often*.

With these assumptions, several aspects have been improved to gain in computational efficiency:

- **Faster visible edges computation.** In order to generate more realistic paths, the obstacle polygons are *inflated*, so the points of the path are slightly detached from the polygon edges. The advantage of this separation in the case of pathfinding, is that the edges of the obstacle polygons are now inside the rooms boundaries, and the number of possible reachable nodes from the agent's location decreases (only the nodes inside the room's boundaries are considered).
- **Obstacle polygons reduction.**
  The obstacles polygons that intersect can be merged, forming a bigger polygon. If most of the walls can be merged (as stated in assumption 3), the number of obstacles is drastically reduced (in the case of the Faculty of Computer Science, the number of wall obstacles are reduced from 2351 to 171, less than 8% of the original) and consequently also the number of intersection tests.
- **The complete path is not needed at once.** The assumption 2 implies that it is not necessary to compute always the whole path between two points A and B. It is frequent, due to events in the environment, that the agent decides to change its current targets before it has reached the end of the path. To avoid unnecessary calculations, at the beginning of the simulation, a navigation graph based on door-room connections is created, and the doors are converted into waypoints. When the agent requests a path, the A* algorithm runs only in the current room.

3.2 Elements localization

An intuitive way for storing the locations of the elements present during the simulation is using uniform grids. However, uniform grids are useful when the spatial data is distributed in an homogeneous way, which is rarely seen during simulation. The use of such grids can easily degenerate into situations where there are many redundant sparse cells. Furthermore, depending on the required accuracy, they can consume too many resources.

People, sensors and actuators need real time information about the elements that surround them. This implies that, during simulation, lots of query ranges must be performed. In order to minimize the impact of this calculation, MASSIS uses a QuadTree [19], a variable resolution data structure for retrieving agents' neighbours within a radius in an efficient manner (see Figure 3). Although some CPU time is required in order to update the structure with the change of each agent, the gain obtained is worth it.

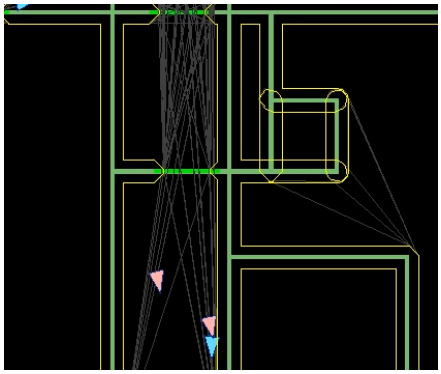

Fig. 2 Visibility graph and merged walls layer

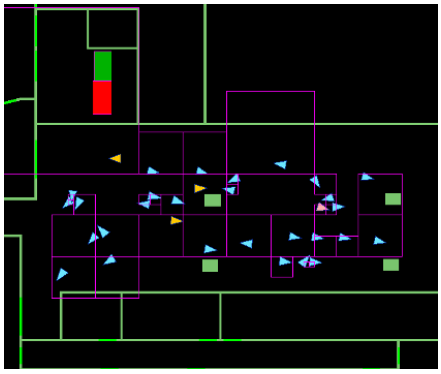

Fig. 3 Agents' Quad Tree layer, showing the space partitioning.

3.3 Steering behaviours

When the goals of the agent have been determined and the path to the target is known, the agent can start moving in the environment, avoiding collisions with obstacles (e.g., other agents, walls, etc.). These basic skills of the agent can be described with *steering behaviours* [2], which need as parameters the agent's mass m, the agent's location $\mathbf{L}$, a maximum force f_{max} and a maximum speed s_{max}.

Every step n, the computed steering forces are applied to the agent's location (limited by f_{max}), producing an acceleration whose magnitude is inversely proportional to the vehicle's mass.

$$\mathbf{A}_n = \left(\frac{trunc(\mathbf{F}_n, f_{max})}{m} \right) \tag{1}$$

The velocity of the agent in every step n is approximated by the Euler integration. Adding the velocity at the previous step ($\mathbf{V}_{n-1}$) to the current acceleration($\mathbf{A}_n$), produces a new velocity:

$$\mathbf{V}_n = trunc(\mathbf{V}_{n-1} + \mathbf{A}_n, s_{max}) \tag{2}$$

Finally, the velocity is added to the agent's location.

$$\mathbf{L}_n = (\mathbf{L}_{n-1} + \mathbf{V}_n) \tag{3}$$

MASSIS provides a flexible implementation of several behaviours of this type (e.g., seek, arrival, separation, collision avoidance, wall containment, and path following, see Figure 4), that can be grouped into more complex behaviours, like flocking or queuing.

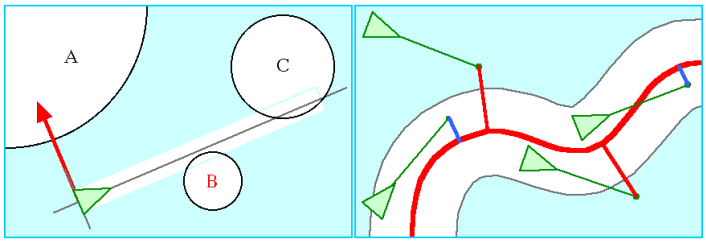

Fig. 4 Some of the steering behaviours implemented in MASSIS framework: Seek and flee, obstacle avoidance and path following.

4 Simulation of the evacuation of a building

It is common practice in public buildings to define some emergency protocols, which may involve, for instance, evacuation of the building. Planning and testing these protocols might be costly, but making simulations about these situations can help to this task (at least, as a first approach). This case study addresses this kind of situation for the building of the Faculty of Computer Science at UCM, which is represented in Figure 5.

Fig. 5 MASSIS 3D representation vs. a real photo

Three kinds of roles have been modelled based on the behaviours described by Proulx [17]:

- **Students** have some knowledge about the building. Their priority is the evacuation, but if they see someone needing help, they will try to assist. They also pay attention to the evacuation signals and indications displayed in the Faculty's CCTV. Their behaviour can be modelled as a POSH plan (see Figure 6).

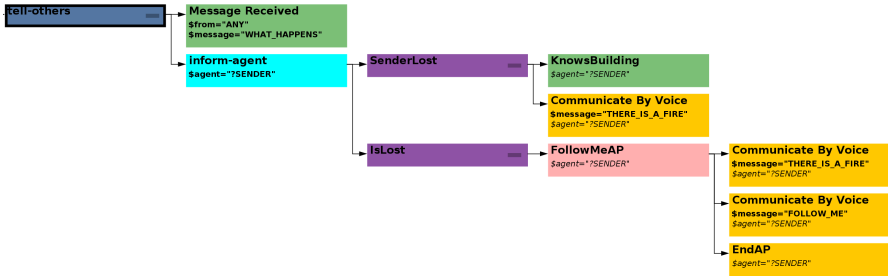

Fig. 6 Partial view of a student's POSH plan, having a drive element (*blue*),a competence(*cyan*), competence elements (*purple*), actions(*yellow*) and an action pattern (*pink*), which models the way in which the agent communicates with other people during the evacuation, reacting differently depending on the characteristics of the other agent. **Note**: some elements were omitted for clarity.

- **Well-trained staff members** are persons who work in the faculty (like a professor or administrative staff). They give instructions to the non-trained people, and try to assure that the evacuation is being done properly, following the established protocol.
- **Visitors** represents persons who have never been in the faculty, so the building is unknown to them. They interpret the fire alarm as something that is happening, so they will start searching for any person, expecting someone to tell them what to do, if something serious is really happening. TV messages can help them to understand that they must evacuate the building, and also other people (an *Student* or a *Well-trained staff member*).

Elements of the environment (sensors and actuators) can be also modelled as agents, with their respective plans, which are usually simpler than those of agents representing people. For instance, Figure 7 shows a fire detector's plan: the existence of a fire triggers its only action, which is the activation of the fire alarm.

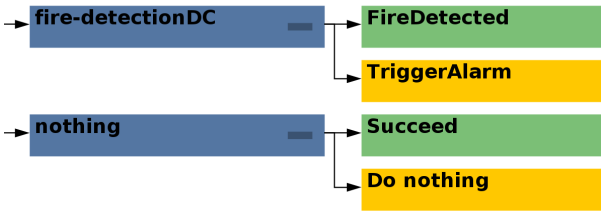

Fig. 7 Simpliest reactive plan: A fire detector.

These behaviour definitions suggest that the agents in this context must be capable of using their visual perception of the environment, what they hear and the ability of interacting with other agents, in order to accomplish higher-level goals. These abilities are modeled using low-level behaviours, managed by POSH primi-

tives, which are the leaves of any reactive plan tree. Figure 6 illustrates parts of the reactive plans used in this case study for the people.

The simulation offers the option to work with 3D and 2D views. Usually 2D views are more practical for analysing what is happening in the simulation. For instance, Fig. 8 represents the state of the agents as colors. Other possibilities are 2D representations of crowd density, perception area, agent's IDs, paths, states, steering forces, etc. Other customized views can be easily integrated.

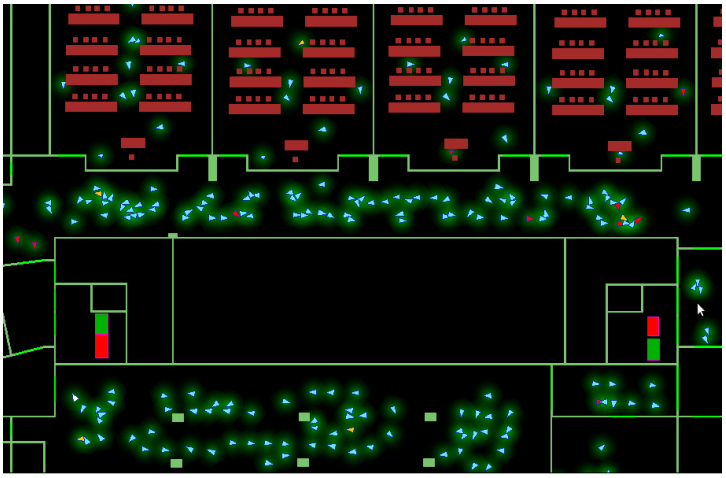

Fig. 8 Simulation 2D view, representing the different state of the agents by color.

5 Conclusions

As it has been shown, MASSIS allows for the efficient simulation of indoor scenarios without losing the ability to specify rich and heterogeneous agent behaviours. This is achieved by simulating each agent individually, but with the support of several methods that take advantage of particularities of the indoor domain.

The extensibility of the MASSIS platform is well supported through its component-based architecture. For instance, different visualizations can be managed during the simulation, new algorithms and agent attributes can be supported and monitored. Also, simulation is logged in order to be able to replicate it and facilitate further data processing to analyse models in more detail.

Acknowledgements This work has been been supported by the Government of the Region of Madrid through the research programme MOSI-AGIL-CM (grant P2013/ICE-3019, co-funded by EU Structural Funds FSE and FEDER), and by the Spanish Ministry for Economy and Competi-

tiveness, with the project Social Ambient Assisting Living - Methods (SociAAL) (grant TIN2011-28335-C02-01).

References

1. Schuerman, M. et al. (2010). Situation agents: agent-based externalized steering logic. Journal of Visualization and Computer Animation 21(3-4): 267-276
2. Reynolds, C. W. (1999) Steering Behaviours For Autonomous Characters. In: Proc. Game Developers Conference 1999, San Jose, California. 763-782.
3. Bryson, J. 2001. Intelligence by design. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.
4. Gemrot, J. et al.: Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents. Agents for Games and Simulations, LNCS 5920, 2009, pp. 1-15.
5. Luke, S. et al. (2005) "Mason: A multiagent simulation environment." Simulation 81.7 : 517-527
6. Treuille, A. C., et.al. (2006) Continuum Crowds. ACM Transactions on Graphics 25(3). Proceedings of SIGGRAPH 2006. 1160-1168.
7. Pan, X. et al.: A Multi-Agent Based Framework for the Simulation of Human and Social Behaviours during Emergency Evacuations. AI & Society, 22 (2007) 113-132
8. "Massive Software Simulating Life." 2002. Accessed Mar. 2015 http://www.massivesoftware.com/
9. Serrano, E., Botia, J. (2013). Validating ambient intelligence based ubiquitous computing systems by means of artificial societies. Information Sciences 222: 3 - 24
10. Algeria l.saifi et al.(2013): Approaches to Modeling the Emotional Aspects of a Crowd.
11. Wu S, Sun Q (2014) Computer Simulation of Leadership, Consensus Decision Making and Collective Behaviour in Humans. PLoS ONE 9(1): e80680. doi:10.1371/journal.pone.0080680
12. Tibor Bosse et al.: Modelling Collective Decision Making in Groups and Crowds: Integrating Social Contagion and Interacting Emotions, Beliefs and Intentions. 6443
13. Bicharra, A. C. et al.(2013) Multi-agent simulations for emergency situations in an airport scenario. Advances in Distributed Computing and Artificial Intelligence Journal 1(3) : 69-73
14. Legion — Science in Motion. Accessed Mar. 2015 http://www.legion.com
15. Galea, E. et. al. (1996). The EXODUS evacuation model applied to building evacuation scenarios." Journal of Fire Protection Engineering 8.2 : 65-84
16. PedGo - TraffGo HT. 2006. Accessed Mar. 2015 http://www.traffgo-ht.com/
17. Proulx, G. (2001). Occupant behaviour and evacuation. In Proc. 9th Int. Fire Protection Symposium. 219-232).
18. Pathfinder - Thunderhead Engineering. 2006. Accessed Mar. 2015 http://www.thunderheadeng.com/pathfinder/
19. Finkel R.A., Bentley J.L. (1974). Quad Trees : A Data Structure for Retrieval on Composite Keys. Acta Informatica 4. 1-9.

Building Prototypes Through 3D Simulations

Jorge J.Gómez-Sanz, Marlon Cardenas, Rafael Pax, and Pablo Campillo

Departamento de Ingeniería del Software e Inteligencia Artificial,
Facultad de Informática,
Universidad Complutense de Madrid, 28040 Madrid, Spain
{jjgomezm,marlonca,rpax,pabcampi}@ucm.es

Abstract. The demo introduces a fast prototyping method based on 3D simulations and the kit AIDE. The system is part of a development philosophy called Virtual Living Lab, where part of the experimentation is made against simulated 3D worlds which show a behavior close to the real one. The demo will illustrate the capabilities of this development philosophy applied to two basic functions working into two different environments: a house and a large installation, such as a faculty building or a mall center.

Keywords: Multi-Agent based simulation, crowd simulation, Inferring population composition, people tracking

1 Introduction

The development of Ambient Assisted Living (AAL) systems is a challenging one. Its is inherently expensive because of, among others, the necessary experimentation and availability of hardware, the uncertainty in the determining the necessary functionality which leads to longer development periods, the need of incorporating the end-user in the development so as to have early feedback, and, at the same time, the complexity of preserving the end-user dignity, privacy, and integrity.

By transferring the development work of the AAL solution to the computer, several benefits are achieved. The main one is the cost reduction since the development environment is not a demanding one and computers designed for gaming will be sufficient. A secondary one is the easiness of experimentation, with the capability of running batteries of tests in a deterministic way. This development philosophy is what we call *Virtual Living Lab*

This transfer has to exist in both directions. It is not only bringing the problem to the computer. It is also being capable of taking out a working solution from the computer. Both transfers are challenging because they imply reproducing the hardware inside the simulation in a convincing way. Past works focused on the Android platform, which was a limiting decision. This iteration of this solution will open the development environment to linux based platforms with enough sensor/actuator availability.

The demo uses software which now is part of the Ambient Intelligence Development Environment (AIDE), which is distributed as free software from

`http://grasia.fdi.ucm.es/aide`. This work bases on previous results, some of them defended in PAAMS [1][3].

2 Main purpose

The demo intends to prove the feasibility of a development using these methods using two platforms: Android, and Beagle Bone. The first platform corresponds to the initial work in this line [1] [2], which targeted Android as deployment platform. This decision was based in the widespread adoption of android technology in most homes and looking for reusing existing hardware in most homes. Nevertheless, this strongly limits the application on other hardware platforms more popular of the Internet of Things, such as Arduino, and reduces the number of sensors/actuators available only to those of Android. Hence, for realizing the *Virtual Living Lab* concept, more hardware platforms need to be considered.

Achieving this goal is not trivial. The variety of operating systems in embedded hardware is appealing. An extensive coverage is too ambitious, so, instead, the goal is to target linux based devices, which are cheap to create and to host, while keeping computing power. Enough power to run some of the open platforms for AAL, such as UniversAAL.

From existing linux based platforms, Beagle Bone (BB) has been chosen for this project. The reason is being BB open hardware and having an extensive collection of pluggable sensors and actuators. This makes this platform suitable for quick prototyping AAL devices with different configurations.

3 Demonstration

The demo consists in the elaboration of example systems oriented towards aiding the user using a variety of sensors. The demo bases on the extensive use of Beagle Bone (BB) for showing several use cases of interest for people with special needs. The demonstration will show cooperation among two or more devices in the simulation and also in the real world, as well as deployment capabilities that enable developers to move software in both directions: from the simulation to the BB devices and vice-versa. The developed functionalities for this demo will cover the following:

- Recognition of activities. The demo will show learning the activities in the simulated world, preparing a control based on this knowledge, and testing this control both in the simulated and the real world. In the demo, the character in the simulation will perform a number of actions with sufficient variations. This will serve to learn and test activity recognition for simple situations.
- Guidance to the user. Through displays, capacitive buttons, and speakers, the user will receive assistance about the activities being performed. In particular, the user is reminded the need of going on with a previously planned scheduled adapted to the user daily activities. This reminder is important for

people with diseases affecting their cognitive abilities, such as Alzheimer's disease. Also, if the user does not follow the advices, it will decide whether to take further action, for instance, contacting relatives or a caregiver, just in case the person is having a disorientation episode.

These two basic functions will be applied in two different scenarios. One will be a house where the user develops a particular daily living activity. The other will be a large installation, such as faculty building or a mall centre, where the user needs assistance to not loose the focus.

4 Conclusions and Reference

The development of AAL solutions is an expensive one, but approaches such as the *Virtual Living Lab* may help to reduce development costs as well as increasing the quality of the developed solution. So far, this was achieved for the Android platform, though this platform has limited possibilities in terms of available sensors and actuators. To overcome this limitation and facilitate the widespread adoption of this philosophy, this demo will prove the feasibility of this approach to more generic linux based platforms. In particular, this demo will focus on the Beagle Bone platform, which is an open hardware platform with an extensive availability of sensors and actuators.

Acknowledgements

We acknowledge support from the project "SOCIAL AMBIENT ASSISTING LIVING - METHODS (SociAAL)", project "Collaborative Ambient Assisted Living Design (ColoSAAL)", and mobility grant grant EEBB-I-15-10097, supported by Spanish Ministry for Economy and Competitiveness, with grant TIN2011-28335-C02-01 and TIN2014-57028-R respectively; and MOSI-AGIL-CM (S2013/ICE-3019) co-funded by Madrid Government, EU Structural Funds FSE, and FEDER.

References

1. Campillo-Sanchez, P., Gómez-Sanz, J.J.: Agent based simulation for creating ambient assisted living solutions. In: Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection - 12th International Conference, PAAMS 2014, Salamanca, Spain, June 4-6, 2014. Proceedings. pp. 319–322 (2014)
2. Campillo-Sanchez, P., Gómez-Sanz, J.J.: A framework for developing multi-agent systems in ambient intelligence scenarios. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015. pp. 1949–1950 (2015)
3. Gómez-Sanz, J.J., Sanchez, P.C.: Achieving parkinson's disease patient autonomy through regulative norms. In: Trends in Practical Applications of Agents, Multi-Agent Systems and Sustainability - The PAAMS Collection, 13th International Conference, PAAMS 2015, Salamanca, Spain, June 3-4, 2015, Special Sessions. pp. 97–104 (2015)

A Greedy algorithm for reproducing crowds

Rafael Pax and Jorge J.Gómez-Sanz

Departamento de Ingeniería del Software e Inteligencia Artificial,
Facultad de Informática,
Universidad Complutense de Madrid, 28040 Madrid, Spain
rpax@ucm.es
jgomez@fdi.ucm.es

Abstract. The gathering of crowd traffic data either from videos or from visual observation has different uses. In the social simulation context, one of them is validating crowd behavior models and match the resulting traffic in control points with the real ones. When such models have been already validated, the immediate use can be aiding managers of facilities to infer, from real time data, what crowd behavior they should expect in their facilities. However, the transformation of those measurements into actual behavior patterns has not been satisfactorily addressed in the literature. In particular, most papers take into account a single measurement point. This paper contributes with an algorithm that produces possible populations that reproduces real traffic data obtained from multiple measurement locations. The algorithm has been validated against data obtained in a real field experiment.

Keywords: Multi-Agent based simulation, crowd simulation, Inferring population composition, people tracking

1 Introduction

The simulation of large groups of people falls within the domain of crowd simulation. Literature gathers examples of theoretical models whose performance is compared with real data obtained from real world situations, such as [6] or [2]. In most cases, it is a validation activity what the researcher tries to achieve. However, there are cases where the problem consists in guessing what kind of population does fit into some observed data. Many populations can exhibit the expected behavior, but not all of them show a believable behavior. Here, believable stands for *satisfying the expectations of the observer*, e.g. an expert such as the social scientist. Generating believable populations is not trivial because the assistance of experts in human behavior must identify the criteria for distinguishing one from the other. Every iteration requires simulating, and then, observing the resulting behavior to decide if the simulation is satisfying. If not, a new population, selected with a different criteria, is needed.

In this open problem, a solution for quickly obtain populations is desirable. With the higher goal of reproducing the behavior of a large group of people, a first step constitutes *producing a traffic data over the control points close to*

the observed one. This problem is relevant for replicating real life situations where the researcher does not have measurements of the whole facility where the measurement is performed, just a portion of this. Validation of the generated population can be made with the *take one out* technique used in machine learning. This technique consists in not using all of the data for training and keep some of it as oracle for assesing the goodness of the population.

This paper contributes with an algorithm for obtaining populations of actors that reproduce traffic data that includes number of people per minute in several zones of a facility. This algorithm is a first step towards creating more complex population that satisfy the criteria of experts in the behavior of large groups. Obtaining such population is not trivial, even if, initially, only traffic data is accounted. Simulated characters remain in the simulation for a long time, not only appear and disappear. Besides, there maybe obstacles, collisions, and wrong estimation of distances that can lead to deviations in the behavior of the population with respect to the observed data. The algorithm included in this paper has a root mean square error of 0.94510658 over observed data.

The structure of the paper is as follows. Section 2 introduces relevant work related to the algorithm introduced in this paper. Section 3 introduces the setup of the problem the algorithm addresses and justifies the complexity of the problem. Section 4 introduces the algorithm itself using pseudocde. Section 5 presents and discusses a sample experiment with a piece of a real field experiment where observations were made. Finally, section 6 contains the conclusions and future work.

2 Related work

Due to their multiple applications, such as sociology, safety analysis, computer graphics or civil engineering simulations of virtual pedestrian crowds have gained increasing attention from industry and researchers. During the last years, the research on data-driven approaches for creating simulated pedestria crowds has been increased significantly, and multiple models have been proposed for solving this issue.

K.H Lee et.al [3] developed a data-driven method of simulating a crowd of virtual humans, extracting trajectories of each individual from the video recordings of an aerial view of a crowd, and learning an agent model from the *state-action* trajectories acquired from visual tracking, having as a result a virtual crowd behaving in a similar way in the same area where the recordings were produced. Following the same paradigm, Lerner et.al [4] propose the creation of an example database for evaluating simulated crowds based on videos of real crowds. Bera. et.al [1] also developed a behavior-learning algorithm for data-driven crowd simulation, capable of learn from mixed videos. Zong et.al [7] developed a framework for generating crowds for matching the patterns observed on video data,taking into consideration the behavior both at the microscopic level as at the macroscopic level. Yi Li et.al [5] developed a technique for populating large environ-

ments with virtual characters, cloning the trajectories of extracted crowd motion of real data sets to a large number of entities.

3 Overview

In this paper, we propose a way for recreating a crowd flow through a multi-agent simulation, where people are modeled as agents, which satisfies the data gathered during the experiment. That is, the people crossing the control zones during the simulation should match with the flow table provided. One of the characteristics of this problem is that the data gathered in the experiment does not contain the information about individual trajectories, only global numbers. This lack of information makes the problem challenging. As the information per person is not provided, the process consists of two phases: Individual path computation and simulation of the crowd flow in a realistic 3D environment.

3.1 Definitions

Control zone We define a *control zone* C as any zone in the environment where the crowd density has been measured over time. Normally, it is a room, a corridor, or an area around a door. Figure 2 illustrates a set of control points placed in an environment.

Time interval	FS_0	FS_1	FS_2	FS_3
$[0, 60)$	6	7	5	3
$[60, 120)$	5	3	1	5
$[120, 180)$	6	1	6	9
$[180, 240)$	1	0	2	1
$[240, 300)$	4	-	3	8
$[300, 360)$	4	-	1	6
$[360, 420)$	5	-	3	4
$[420, 480)$	6	-	5	3
$[480, 540)$	5	-	1	5
$[540, 600)$	6	-	6	9
$[600, 660)$	1	0	2	1
$[660, 720)$	4	0	3	8

Table 1. Example of a flow table

Time record The number of people that have entered a control zone C_i during a concrete time interval is represented by a *time record*, which consists on a triple $T_i = \{t, n, C_i\}$, where n is the number of people that have crossed the control zone in the time interval $t = [t_{start}, t_{end}]$. The value of t is expressed as

an interval because depending on the people counting method, this data cannot be measured with exact precision (people counted by a human observer, for example).

Flow set All the time records belonging to a particular control zone C_i are grouped in a flow set $FS_i = \{T_{i_0}, T_{i_1}, \ldots, T_{i_n}\}$. The natural way for representing this data is arranging the time records in a table, (called *flow table*) being in the same row the ones having the time intervals in common. Table 1 illustrates how this arrangement is done. An entry containing a hyphen (-) instead of a number means that there is no data available for that time interval in the control zone (perhaps it was collected with different time intervals, or because the observation of the people flow was not performed during that time).

4 The Algorithm

Due to the lack of information about individual trajectories, different configurations for each agent should be generated in order to create a set of individual trajectories matching the data collected. As the recreation may involve many control zones, and large periods of time, we have developed a greedy approach in order to solve this problem. This approach creates trajectories passing through the control zones, taking into account the distances between them and the speed of each agent. It is assumed that there are no agents at the beginning of the simulation, that they come from outside of the environment via some specified *entrance points*, and that they exit the environment through another entrance point. The trajectories are computed in a reverse way: The first point added to the trajectory is the last one the agent will pass through. At each iteration, a set of reachable control zones is selected by a *selection function*, which takes into account the agent's speed and the the current control zone of the agent. Then, the number of people in the selected control zone is decreased by 1. The process is repeated until the first time interval is reached.

Listing 1 Main Algorithm

```
while Flow table is not empty do
    endLoc ← selectEntrance()
    tr ⟵ the time record with maximum time
    tr.n = tr.n − 1
    availableTime ← tr.t
    loc ← tr.c
    agentSpeed ← speedSelection()
    availableTime ← availableTime − travelTime(endLocation, loc)
    trace ← GenerateTrace(availableTime, loc, tr)
    traces ← traces + GenerateTrace(availableTime, loc, tr, speed)
end while
    return traces
```

The algorithm ends when the flow table is empty: If the n value of all the time records is zero, there is no need to generate trajectories.

At the beginning, an *entrance point* is selected. It will be the location where the agent will *exit* from the environment (remember that the algorithm runs in a reverse way). After that, a non-empty time record (tr) with the highest time value is selected. The control zone corresponding to this time record will be the *last* control zone that the agent will cross before exiting the environment. If there are more than one available, can be chosen with a custom criteria.

Listing 2 Trace Generation

```
trace ← InitializeTrace
while availableTime ≥ 0 do
    available ← availableTargets(availableTime, loc)
    if available is not empty then
        tr ← selectTarget(available)
        availableTime ← availableTime − travelTime(c, loc, speed)
        loc ← locationOf(tr)
        tr.n = tr.n − 1
        addToTrace(loc)
    else
        if availableTime > 0 then
            ip = selectIntermediaryPoint()
            availableTime ← availableTime − travelTime(ip, loc, speed)
            addToTrace(loc)
        end if
        if availableTime ≤ 0 then
            go to the start point
            eP ← selectEntrance(B)
            addToTrace(loc, trace)
            availableTime ← availableTime − travelTime(eP, loc, speed)
        end if
    end if
end while
    return trace
```

The n value of the selected time record ($t.n$) is decremented by 1. This is done because if n agents will be passing through that the corresponding control zone in the time interval $t.t$. Doing this operation n times for different agents (or the same agent, if the time interval is large enough), will satisfy the condition of the initial problem. After the initialization, the process of selecting time records is done iteratively, applying the same concept:

- An available time record is selected. A time record tr is "available" if the $tr.n > 0$ and the agent can reach it in the time interval $t.t$.
- If there is no time record available (There are too far from the agent's location, for example), an intermediary point in the environment is selected.

– Every time that a location is selected, the available time is decreased, taking into account the distance from the agent location and the agent's speed.
– When the *available time* is less or equal to zero, an entrance point is selected. This entrance point will be the *first one* that the agent will cross.

4.1 Example

For illustration purposes, let's consider a simple example with three control zones, C_0, C_1 and C_2. For an agent with an specific speed, the travel times in seconds between the control zones could be 200 ($C_0 \rightarrow C_1$), 100 ($C_0 \rightarrow C_2$) and 150 ($C_1 \rightarrow C_2$), as shown in Figure 1. The initial state of the flow table is shown in Table 2.

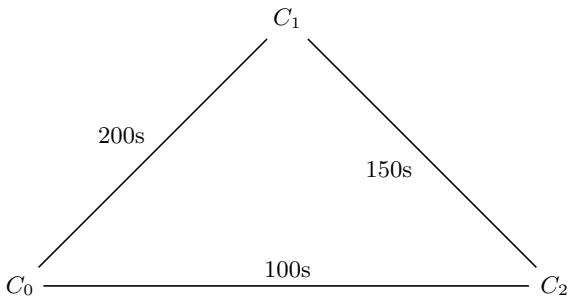

Fig. 1. Travel times from the different control zones of the example

Time interval	FS_0	FS_1	FS_2
$[0, 60)$	6	7	5
$[60, 120)$	5	3	1
$[120, 180)$	6	1	6
$[180, 240)$	1	0	2
$[240, 300)$	4	5	3
$[300, 360)$	4 ←	0	1

Table 2. Initial state.

Time interval	FS_0	FS_1	FS_2
$[0, 60)$	6	7	5
$[60, 120)$	5	3	1
$[120, 180)$	6	1	6
$[180, 240)$	1	0	2
$[240, 300)$	4	5	3 ←
$[300, 360)$	[3]	0	1

Table 3. Second iteration.

When the algorithm starts, an non-empty record having the time interval with the highest value is chosen. In this example the chosen time record shown in table 2. The candidate time records are the ones highlighted, and the chosen time record ($\{[300, 360), 4, C_0\}$) is marked with an arrow (←).

Once the time record is selected, its value is reduced by one, and the location of the agent is saved. The candidate time records after the first step are highlighted in table 3. The time intervals are $[240, 300)$ and $[120, 180)$, because traveling from C_0 to C_2 takes 100 seconds, and $359 - 100 = 259 \in [240, 300)$, being the other case analogous. In this step, the candidate time record that will be chosen will be $\{[240, 300), 3, C_2\}$, because its time interval is higher. In the next iteration, the candidate time records are $\{[60, 120), 3, C_1\}$ and $\{[120, 180), 6, C_0\}$.
If we follow the same process that the one done in the previous step, the selected time record should be $\{[120, 180), 6, C_0\}$. But it is obvious that the trajectories of the agents would be awkward, moving forward and backwards every step. This restriction depends on how the control zone selection function is implemented.

Time interval	FS_0	FS_1	FS_2
$[0, 60)$	6	7	5
$[60, 120)$	5	3 ←	1
$[120, 180)$	6	1	6
$[180, 240)$	1	0	2
$[240, 300)$	4	5	[2]
$[300, 360)$	3	0	1

Table 4. Third iteration.

Time interval	FS_0	FS_1	FS_2
$[0, 60)$	6	7	5
$[60, 120)$	5	[2]	1
$[120, 180)$	6	1	6
$[180, 240)$	1	0	2
$[240, 300)$	4	5	2
$[300, 360)$	3	0	1

Table 5. Last iteration

The last step is shown in table 5. After this last step, a path from the control zone chosen (e.g., C_0 or C_2) to an entrance point should be computed.

5 Case study

The case study considers the floor of a building,(see Figure 2), where control areas, painted in red, have been defined. Using this blueprint, samples of traffic data through red areas is used as input. The number of people crossing that control zones was annotated with a frequency of one minute.
The crowd behavior consists of entering the building through designed entrances and move along the floor in a way that traffic data through red areas matches input data. Data used for the experiment comes from a real human traffic observation in a configuration similar to the one depicted in Figure 2, where the control zones are marked as red circles, and entrance points are marked as blue squares.
The traffic of the inhabitants during the simulation is represented in Figure 3. The characters move along different pathways automatically generated. These pathways are designed in a way that, assuming a constant

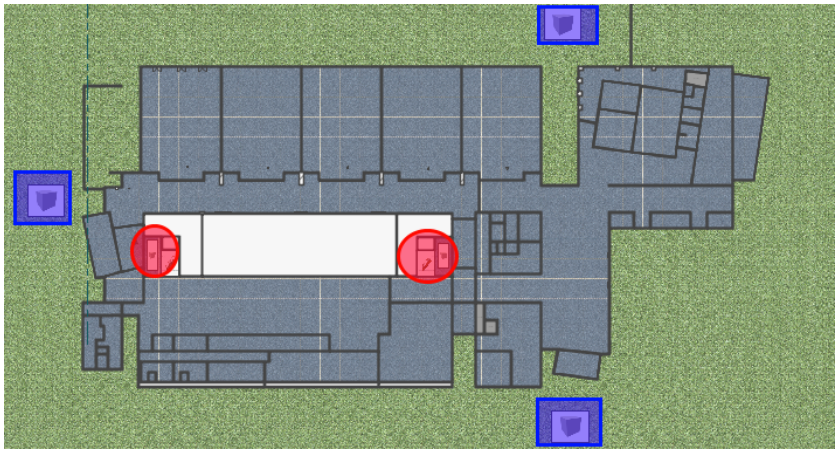

Fig. 2. Control zones of the case study.

speed, the characters cross the intended red areas at the designated times. The pathways may look unnatural because of this limitation. Hence, one can observe a character going to the middle of a room and just returning. One of the future work improvements consists in adding activities to perform along the pathways so that the simulation is more realistic. Despite this limitation, the obtaining traffic data from the simulation is close to the ideal.

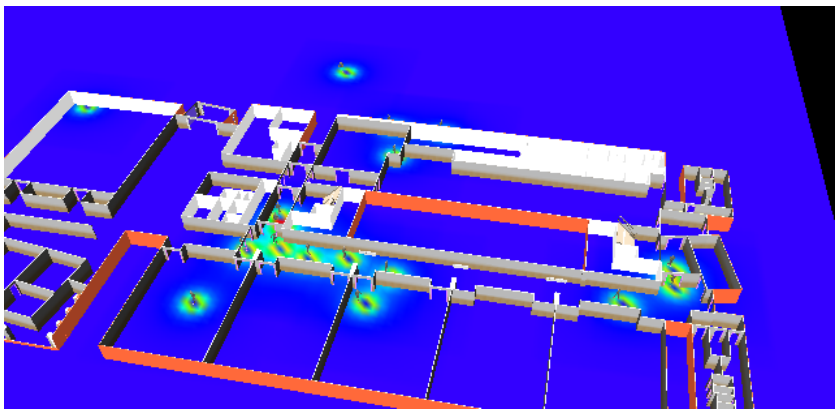

Fig. 3. Sample execution of the simulation over the blueprint from figure 2

Some of the observed results during one execution are presented in figure 4 and in tables 6 and 7. It can be observed a difference between the number of people that passed over the control zone in the real measurements and the simulated ones.

Time interval	Real	Simulated
0	4	10
60	1	4
120	4	9
180	7	9
240	2	6
300	8	14

Table 6. Control zone A density values

Time interval	Real	Simulated
0	10	14
60	1	2
120	6	9
180	1	5
240	5	12
300	2	3

Table 7. Control zone B density values

Two reasons may explain these results. First, the algorithm does not take into account (yet) the movements of other agents during its execution, the collision detection between agents and its resolution is done during the the simulation, not before. Second, that a trajectory may cross an undesired control zone, making the results slightly worse. The plot reported in Figure 4 shows how this issue alters the simulation results, increasing the number of people counted over one of the control points of the simulation.

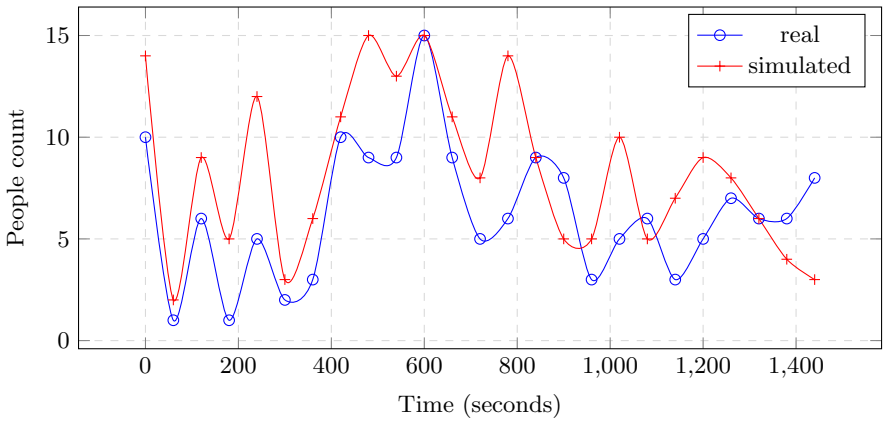

Fig. 4. Control zone crowd density values, real vs simulated.

6 Conclusion and future work

This paper has introduced an algorithm for creating populations of characters whose behavior produce a traffic data close to observed data. This population is made of characters that navigate through a facility which has

obstacles and where characters can collide. The algorithm fails to give precise results so as to remain domain independent. Hence, there are differences in the navigation between the simple behavior assumed by the algorithm and the obtained behavior for the simulation. This difference is not high, but it may increase according to the complexity in the environment. Also, the algorithm does not account activities of the daily living of the characters beyond a navigation path going on for a long time and that goes through selected control points.

The relevance of the algorithm for research in crowd simulation is for creating specific populations that show specific traits in their behavior, such as traversing control points at precise times in a long simulation. Such populations can be used to validate theoretical models of crowd behavior. The number of populations that satisfy the constraints has not been studied. If only a few are possible, this may be an indirect way of determining the behavior of large groups when only a few observation points are available.

Future work will take into account the complexity of the environment as well as in the behavior of the characters in a way that the resulting behavior is closer to the observed behavior of large groups of people.

References

1. Bera, A., Kim, S., Manocha, D.: Efficient trajectory extraction and parameter learning for data-driven crowd simulation. In: Proceedings of the 41st Graphics Interface Conference. pp. 65–72. Canadian Information Processing Society (2015)
2. Berrou, J.L., Beecham, J., Quaglia, P., Kagarlis, M.A., Gerodimos, A.: Calibration and validation of the legion simulation model using empirical data. Pedestrian and evacuation dynamics pp. 167–181 (2005)
3. Lee, K.H., Choi, M.G., Hong, Q., Lee, J.: Group behavior from video: A data-driven approach to crowd simulation. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. pp. 109–118. SCA '07, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2007), http://dl.acm.org/citation.cfm?id=1272690.1272706
4. Lerner, A., Chrysanthou, Y., Shamir, A., Cohen-Or, D.: Data driven evaluation of crowds. In: Motion in Games, pp. 75–83. Springer (2009)
5. Li, Y., Christie, M., Siret, O., Kulpa, R., Pettré, J.: Cloning crowd motions. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. pp. 201–210. Eurographics Association (2012)
6. Rivers, E., Jaynes, C., Kimball, A., Morrow, E.: Using case study data to validate 3d agent-based pedestrian simulation tool for building egress modeling. Transportation Research Procedia 2, 123–131 (2014)
7. Zhong, J., Cai, W., Luo, L., Yin, H.: Learning behavior patterns from video: A data-driven framework for agent-based crowd modeling. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. pp. 801–809. International Foundation for Autonomous Agents and Multiagent Systems (2015)

Towards the Simulation of Large Environments

Jorge J. Gomez-Sanz and **Rafael Pax** and **Millán Arroyo** [1]

Abstract. The development of a smart environment working into large facilities is not a trivial matter. What kind of intelligence is needed and how this intelligence will interact with individuals is a critical issue that cannot be solved just by thinking about the problem. A combination of social and computer science methods is necessary to learn and model the interplay between the environment and the environment inhabitants. This paper contributes with an ongoing case study that exemplifies this kind of combination. The case study considers two faculty buildings and a behavior to be modified. The goal is to design a set of devices that sends signals to passing-by pedestrians in order to make them use more the staircases. Banners, videos, and directed intervention are used. The effect of each one is measured and such measurements are reproduced into computer simulations. This information is necessary in order to determine the duration, the intensity of the stimulus, and the response of the individuals. Opposite to most works, the measurements do not provide full information of what is going on in the large facility. As a consequence, algorithms and software to fill in the gaps consistently are needed. The paper describes the current state of the simulations and the difficulties in modeling with precision the results in a case study.

1 Introduction

If a large facility is expected to host an embedded system, as in a Internet of Things or Ambient Intelligence scenario, the definition of such system and its early validation is largely missing in the literature. Given a particular building or large space, a first question is whether the goal to make the visitors of the facility show a new behavior or to alter an existing one. This cannot be done in the traditional way, by consulting a few stakeholders. Interviewing and surveying the occupants of the environment seems more adequate.

How this information is captured and reused later on, it is still an issue. Documenting is out of question. However, the format of the documentation can be subject of discussion. Also, how this documentation is used and accessed in order to ensure the problem is completely understood.

The hypothesis of this work is twofold. First, that social sciences methods can be used in order to capture better the behavior of the humans inhabiting the large facility and what stimulus can trigger this behavioral change. Second, once there is a preliminary solution to the creation of new behaviors or modification of existing ones, the enactment of such behaviors can be something better documented if computer simulations are used.

Besides, documenting something as dynamic as the behavior of big groups of humans through simulations allows too to experiment with the expected effect of different stimulation procedures. This way, responsible of large facilities can have tools that enable them to discover how they want the facility to be altered before the actual smart environment is even built. In order for the simulation to be realistic, the simulation has to reproduce the interplay among users and between users and the environment. The later includes too the devices that are supposed to make the users behave differently. These devices do provide stimulus previously validated by experts as candidates to produce the desired kind of effects.

The paper is structured as follows. First, section 2 analyses if a particular behavior alteration/induction is really possible. The contribution of social sciences to the requirements gathering is made in 3. A guideline is proposed that combines interviews, surveys, but also field studies, as well as analyses of the captured information. Examples of the analysis phase is made in section 4, where a domain example is introduced. The design of simulations that aid to create the smart environment capable of enacting the new behaviors is made in section 5. The related work is introduced in section 6. Conclusions are presented into section 7.

2 Stimulus for Behavior Alteration

Literature shows that humans are sensible to external stimulus in subtle ways and that our behavior can be altered. The extent of the alteration may depend on the individual. Some may react notably while others hardly react. Nevertheless, the average person ought to notice this. The nature of the stimulus matters too. In certain conditions, such as evacuations, humans pay more attention to other humans rather than other artificial elements, such as banners.

Humans have sensibility towards the behavior of other humans. If an individual finds a group along the way, depending on its size, will either stop and look what is happening and stay or keep walking [11]. The larger the group, the greater the effect. This is explained as a mirroring behavior effect. If sufficient people stare at an arbitrary point, a passerby individual will unconsciously look at the same place [5]. Gaze copying happens mainly within 2 meters range and the response depends on the physical layout of the environment, the social context, and the sex of the individual.

When the stimulus come from artificial sources, the results are still promising. Sound and images can affect the behavior of pedestrians. Beyer et al. [2] introduce an experiment where an interactive large banner display affects the audience. Through visual stimulus, authors manage to attract approaching pedestrians and distribute them along the display. Miller [12] shows that noise can affect people's performance. A sleepy person may be aroused by noise, but it has also negative effects, like affecting the performance of complicated tasks, affect negatively the mood and disturb relaxation. Negative effects could be used to influence pedestrians. In this paper, it is assumed that, since it can annoy people, this could be used to clear out areas or to reduce the pedestrian traffic around some places where the noise comes from.

The context matters too. Foster [4] analyzes different domains in

[1] Universidad Complutense de Madrid, email:jjgomez@ucm.es

order to promote healthy habits. Each context is different. A shopping center and railway station involve different behaviors on behalf the population.

Also, sensibility towards stimulus changes depending on the context. In an airport, passengers pay special attention to information panels. A change in one panel may trigger movements of user groups, such as changing one boarding gate ten minutes before the boarding starts. Fun parks also influence the behavior of their visitors through information panels that tell expected waiting time for each attraction.

3 Guidelines for Developing Ambient Intelligence in Large Facilities

The system to be developed aims to interact with several inhabitants of a large space. These inhabitants may be transient ones or permanent inhabitants of the considered space. It is assumed that the people in this physical environment can be either a management staff, in charge of the facilities and aiding to the occupants of the facilities to fulfill the identified system goals; and the visitors, who are the clients of the facility. In general, the staff interacts with the visitors in order for helping them perform certain activities. In the physical space, it is assumed the staff is expected to modify the behavior of the clients in a way that clients perceive an benefit.

To identify what behavior modifications are possible and how to best convince inhabitants of the facility to commit to such behavior, a guideline affecting particular system development is introduced following:

- Analysis phase. The facility to be analyses is assumed to fulfill one or many purposes. The staff is expected to alter the behavior of the inhabitants in order to achieve certain behavior. This behavior is compatible with the purposes of the facility, and it is supposed to be regulated or activated through some environmental devices. There is a review of the meaningful behaviors, according to the literature, on the expected behaviors (domain or non-domain specific) for the chosen facility. A selection of stimulus is made based on the available resources (the budget of the modification, for instance). Also, field studies have to be planned to know more of the visitors and also to evaluate the effect of those stimulus over time. Effect of each stimulus is measured and annotated so that it can be reproduced later on. Each stimulus is expected to have a duration and an intensity.
- Design phase. The different stimulus and the expected reaction is modeled into a simulation that serves as reference. The simulation includes the physical space, the inhabitants of the space, the expected behaviors of those inhabitants according to the field studies, and the simulated devices that are going to provide the stimulus. The measurements made in the field studies are interpolated to guess the overall behavior of the whole population. Accordingly, the expected behavior is studied, taking into account the reaction to the stimulus. As a result, an expected orchestration of the stimulus is obtained.
- Deployment phase. The synchronization of the stimulus is deployed into real devices already working in the facility. The simulation is expected to have identified several critical observation points whose measurements indicate if the stimulus is working or not.

The role of human scientists is important in the development of this kind of systems. In this guideline, it is assumed that human scientists involve themselves mainly into the analysis stages. However, their collaboration is needed too along the design stages. Human sciences scientists, such as psychologists or social scientists, provide insight into the behavior of the users beyond common wisdom. Hence, they are needed in order to properly design the field experiments, to study the results, and to assess the validity of the simulations.

It is assumed that there is a simulation parameterization whose behavior is close to the observed behavior. Such simulation should be possible because the behavior of users into installations is not heterogeneous and tend to fit into standard behavior patterns, that we associate with activities of the daily living typical of the installation. The definition and parameterization of the simulation is considered following.

3.1 Specifying the crowd simulation

An important part of the simulation is the description of the physical space inhabitants, which is called here population description. For this goal, it is necessary to identify a set of possible actor behaviors, an enumeration of the number of instances of these behaviors, and a timestamp of when the behavior&actor instantiation happens.

Actor instances are created along the simulation and destructed when the behavior of the character finishes. It is assumed the designer determines a suitable place where this destruction happens. After all,actors cannot vanish from the scenario just anywhere. These actors instantiate a particular set of behaviors with particular parameters. The different parametrization determines individual variations of the behavior.

It is assumed that actors can belong to two distinguished groups: those responsible of operating the facilities and those visiting the facilities. The first are expected to perform different activities oriented towards coordinating the behavior of the second group within the facility. The second group are executing activities of the daily living related to the main purpose of the facility. It is not expected that one actor belonging to one group suddenly becomes an actor belonging to another. Even thought there maybe cases where this role switch makes sense, it is not considered in this paper. Within each group, there can be further decomposition of responsibilities, but it depends on each particular domain.

An actor behavior specification consists of a sequence of parameterized activities of the daily living plus an initial location. The amount of instances of each actor behavior specification determines the composition of the population.

Actors are not allowed to alter their behavior and they constantly perform the same sequence. The sequence terminates with the destruction of the character. This enforces designers to define precisely what actors are expected to do since their creation until completing their part in the simulation.

4 The case study

The crowd simulation has been applied to a scenario situated in two faculties. The goal is to alter the behavior of the inhabitants in order to make them choose an activity that requires additional effort over another activity that does not. The behavior to be altered is using the elevator, which ought to be replaced by using the staircases. The experiment is run into two different faculties, the Computer Science Faculty and the Political and sociological Sciences faculty.

The application of the methodology starts with a field study structured as follows. First, the managers of both faculties are interviewed to know more of the daily problems they have to face. This provides an insight on the students and other staff using the facility. It also

helps to identify possible incompatibilities between the planned stimulus and the current activities. The chosen stimulus are:

- Human-to-human interaction. A person playing the role *facility operator* interacts with another playing the role *visitor* and tries to suggest the use of staircases is better.
- Banners. Banners are proposed containing information of interest to the visitor and that may aid in suggesting an alternative behavior. It is important to notice that there ought to be an evident profit for the visitor, otherwise the behavior modification will not occur. In this case, the banner is presented at figure 1. It suggests the visitor will gain health improvements, will arrive faster to the destination, and will save electricity. These facts, specially the savings in time during travels, has been proven to be true.
- Multimedia. A video shows a dramatization of a person that uses the elevator for everything even though can perfectly walk. The video is shown through a short distance beamer sufficiently visible and the equivalent of a 55' screen. The short distance beamer is projecting vertically and permits a less disturbing installation. The projection is made close enough to the elevator. Due to safety concerns, it was not allocated right next to the elevator.

Figure 1. Banner for motivating users to use the staircase. It written in spanish. The main title says *stair climbing and avoiding elevators* at the top. The alleged reason are *1. improving your health*, *2. You will get faster to your destination*, and *3. you will save energy*

A plan for measuring the effects of these stimulus was made. The plan consisted on a five week schedule. The first week (week A) there was no stimulus and it was used to collect a base line of staircase/elevator traffic stats; during the second week (week B) the banner stimulus was introduced; along the third week (week C1) the videos were added; and in the fourth week (week C2), the human-to-human interaction. Then, there were some days of no stimulus to let users decide whether they want to keep the new behavior or get back to the old one. Therefore, the fifth week (week D) is dedicated to measure the resilience of the stimulus.

Collected data was a set of pedestrian traffic into strategic checkpoints of the faculties. Measurements indicate whether visitors come or go, and whether they are using the elevator or the staircases. An account of persons per minute is provided. The resulting influence of the stimulus along the field experiment stages is included in table 1. The number of people arriving through the elevator remains mostly the same along stages. However, the number of people choosing not to use the elevator is reduced up to 4 points in phase C2. This is a variation of 13,65% over the original use of the elevator. The results are not shocking, but it should be taken into account that each stimulus lasted for one week, and not months.

Table 1. Variation of the traffic in elevators into two faculties

% use elevators	A	B	C1	C2	D
Total	23,1	21,9	21,4	20,3	22,2
Departures	29,3	28,2	26,2	25,3	26,4
Arrivals	14,4	14,4	15,2	14,0	16,2
#total=	9730	9797	9459	9165	9088
#departures=	5688	5371	5335	5109	5345
#arrivals=	4042	4426	4124	4056	3743

With the obtained traffic data, a simulation is arranged so as to reproduce the observed behaviors.

5 Reproducing the experiments

The result of the experiments is being transferred to computer simulations, to identify complexities and capture individual behavior as precisely as possible.

In the simulation, all actors are belonging to the visitor role. Their actions consists in entering the building, visiting a previously unknown number of rooms, and exiting the building. Hence, a parameterization of the problem includes an account of the rooms each actor visits and how long they stay there.

The computer simulation has to capture emergent behaviors. Rather than organizing dynamically the behavior of a whole population and letting a central node orchestrating everything, the approach is multi-agent based one, where individual behaviors of characters is coded. The individual behaviors is explained along the next paragraphs, but the goal is to attain the same, or close, traffic data to those obtained from the different experiments. Since the data from each phase is available, the simulation ought to capture the effect of the stimulus over the visitors. Henceforth, if the stimulus is a banner and the measured effect is a 25% variation, then the simulated traffic ought to show such change as well.

The total aggregation of the traffic ought to provide with numbers similar to those of table 1. Achieving this traffic data while coding individual behaviors is a hard task because of two reasons. First, there are several elements whose interplay affects the outcome of the simulation. Actors interact among themselves and with the environment, specially elevators and the physical layout of the environment, a building with several floors. Second, the gathered information is partial, since only a few pedestrian traffic check points were established in the field study from section 4. This means there were not cameras recording the full activity. As a consequence, there may be many populations of simulated actors whose movement along the facility matches the obtained measurements in the field experiment of section 4

The problem has been studied in [13] and the provisional solution is a greedy algorithm that produces a population of actors whose behavior matches to some extent the expected behavior of the whole population. A first attempt is presented in figures 2 and 3.

The behavior of each individual can be summarized as follows. Each character has a navigation path from the starting point to a particular location determined by the greedy algorithm [13] and going through some intermediate points that are part of the parameterization. Intermediate points may correspond to specific rooms the

characters may or may not visit. Along the navigation, the character may find obstacles. Fixed obstacles are already avoided by the navigation algorithm. Mobile obstacles are avoided through maneuvers around the expected collision points. Afterwards, the navigation path is rechecked and resumed.

Figure 2 shows a part of the 3D simulation created with the greedy algorithm. In the simulation, to compare the simulated vs the real scenario, the simulation assumes there is a device in the area capable of counting people as they cross the section corresponding to the checkpoint. The counting is compared against the real measured traffic in the bottom part of the figure.

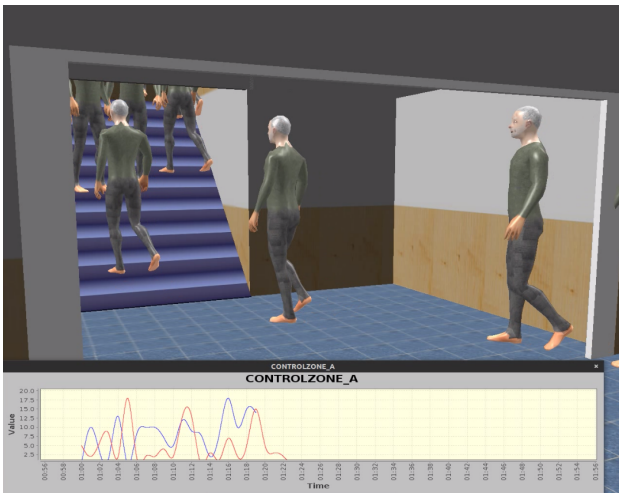

Figure 2. Simulation of pedestrian traffic along checkpoints and simulated traffic data gathering

There are many possible populations of actors whose movements have the same effect in terms of traffic through the checkpoints, at least, in theory. The greedy algorithm from [13] achieves the performance shown in figure 4. This figure focuses on the traffic data and compares the simulated to the real measured traffic along the experiment. The considered time window is different from 2. In figure in 3, there is a small variation in the obtained simulated traffic measurements. The main reason for such variations is the interplay of actors along their paths, which is not taken into account. Collisions and bottlenecks happen too, and they cause a different transit time. This is a positive sign the simulation is more complex than the simplified model the greedy algorithm uses.

Another source of complexity is the modeling of elevators, as shown in figure 3. The characters that occupy the interior of the elevator must coordinate to exit into each floor. Problems happen when one character situated at the back of the elevator wants to get out, but no one of those situated at the front wants to move. Again, this alters the traffic. To prevent this, the simulated actors have to be aware of what is the right use of an elevator.

The problem becomes more complex when the activities of the daily living is added to the considerations. The protocol of lectures in a classroom is simple: students come to the classroom; they sit down; a teacher comes and starts the lecture; more students may come during the lecture; the lecture finishes and then all, or a few, students leave the room. The uncertainty in the process, such as teachers finishing sooner or later, makes the evacuation of students from classrooms more smoother than it should be if all teachers coordinated

Figure 3. Elevator carrying people from one floor to the other

precisely the lectures to finish exactly at the same time. Such individual behaviors are relevant to be modeled too.

Also, the simulations may lead to inconsistent results. For instance, most of the resulting populations according to the algorithm [13] have in common that upper floors are mostly empty. Upper floors only have offices and not classrooms, what would explain this result. Then, it may be subject of discussion if a better occupation of the building was possible. If the space allocated in upper floors is the same as lower floors while the traffic is much lower, perhaps a higher number of offices could be arranged without compromising an eventual evacuation of the building.

Capturing complexity at the simulation allows to realize the software-in-the-loop approach. It is a goal of the project to include sensor/actuator devices in the simulation so that a designer can explore the effect of the stimulus of those devices on the population. The simulated devices would be operated using control software that was close to the simulated one. This approach has been essayed in [6] for gesture recognition devices design using 3D simulated environments generated with the AIDE environment [3].

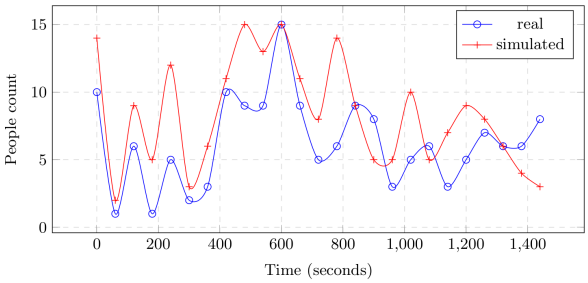

Figure 4. Measured traffic in the simulation compared to observed real traffic using a different time window from figure 2

6 Related work

There are works dealing with the design of smart systems, but they do not frequently consider human sciences and stimulus to plan the kind of system which is needed and what performance it will have. Harrison [7] claims the analysis of mutual and incidental user interaction

has not been accounted and proceeds to apply fluid flow analysis to understand it. This kind of analysis is necessary, but, it does not replace a more conventional study and cannot assume a 100% response of the individuals every time. Other works focus on the devices expected to provide the stimulus at small scale, such as [14]. Thought authors stress the involvement of human scientists too, the behavior of people in small spaces cannot be compared to that of large spaces.

There are precedents too in reproducing observed data as simulations. In [8], video recordings were used to reproduced later on a crowd simulation of simulated actors. Behavior of the individuals were obtained from a multiple checkpoint observation that allowed. The project introduced in this paper, however,assumes incomplete information about activities and traffic. The less information is used, the less expensive a real installation would be. Following the same paradigm, Lerner et.al [9] propose the creation of an example database for evaluating simulated crowds based on videos of real crowds. Bera. et.al [1] also developed a behavior-learning algorithm for data-driven crowd simulation, capable of learn from mixed videos. Zong et.al [15] developed a framework for generating crowds for matching the patterns observed on video data,taking into consideration the behavior both at the microscopic level as at the macroscopic level. Finally, Yi Li et.al [10] developed a technique for populating large environments with virtual characters, cloning the trajectories of extracted crowd motion of real data sets to a large number of entities.

7 Conclusions

The paper has introduced a guideline with recommendations inspired into human sciences and the realization of field experiments. Such experiments are necessary to fine tune devices that aim to influence the behavior of the facility inhabitants. With this information, computer simulations have been created. These simulations reproduce the observed behavior and can be used to experiment with different setups and stimulus until a suitable combination is found. The next step is to devise the control software capable of synchronizing the stimulus over the population and then deploying such software to real devices having that capability.

Simulations can be used also to reason about what is happening inside the facility. In particular, the produced simulations show there are concerns in how the building is actually used. The identified traffic, and assuming most of the people that goes through a staircase/elevator do it only once, permits to infer that the upper floors of the building are less occupied than expected.

ACKNOWLEDGEMENTS

We acknowledge support from the project “Collaborative Ambient Assisted Living Design (ColoSAAL)” (TIN2014-57028-R) funded by Spanish Ministry for Economy and Competitiveness; and MOSI-AGIL-CM (S2013/ICE-3019) co-funded by Madrid Government, EU Structural Funds FSE, and FEDER.

REFERENCES

[1] Aniket Bera, Sujeong Kim, and Dinesh Manocha, ‘Efficient trajectory extraction and parameter learning for data-driven crowd simulation’, in *Proceedings of the 41st Graphics Interface Conference*, pp. 65–72. Canadian Information Processing Society, (2015).

[2] Gilbert Beyer, Vincent Binder, Nina Jäger, and Andreas Butz, ‘The puppeteer display: attracting and actively shaping the audience with an interactive public banner display’, in *Proceedings of the 2014 conference on Designing interactive systems*, pp. 935–944. ACM, (2014).

[3] Pablo Campillo-Sanchez and Jorge J. Gómez-Sanz, ‘Agent based simulation for creating ambient assisted living solutions’, in *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection - 12th International Conference, PAAMS 2014, Salamanca, Spain, June 4-6, 2014. Proceedings*, pp. 319–322, (2014).

[4] Charles Foster and Melvyn Hillsdon, ‘Changing the environment to promote health-enhancing physical activity’, *Journal of sports sciences*, **22**(8), 755–769, (2004).

[5] Andrew C Gallup, Joseph J Hale, David JT Sumpter, Simon Garnier, Alex Kacelnik, John R Krebs, and Iain D Couzin, ‘Visual attention and the acquisition of information in human crowds’, *Proceedings of the National Academy of Sciences*, **109**(19), 7245–7250, (2012).

[6] Jorge J Gómez-Sanz, Marlon Cardenas, Rafael Pax, and Pablo Campillo, ‘Building prototypes through 3d simulations’, *Advances in Practical Applications of Scalable Multi-agent Systems. The PAAMS Collection*, 299, (2016).

[7] Michael D Harrison, Mieke Massink, and Diego Latella, ‘Engineering crowd interaction within smart environments’, in *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pp. 117–122. ACM, (2009).

[8] Kang Hoon Lee, Myung Geol Choi, Qyoun Hong, and Jehee Lee, ‘Group behavior from video: A data-driven approach to crowd simulation’, in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA ’07, pp. 109–118, Aire-la-Ville, Switzerland, Switzerland, (2007). Eurographics Association.

[9] Alon Lerner, Yiorgos Chrysanthou, Ariel Shamir, and Daniel Cohen-Or, ‘Data driven evaluation of crowds’, in *Motion in Games*, 75–83, Springer, (2009).

[10] Yi Li, Marc Christie, Orianne Siret, Richard Kulpa, and Julien Pettré, ‘Cloning crowd motions’, in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 201–210. Eurographics Association, (2012).

[11] Stanley Milgram, Leonard Bickman, and Lawrence Berkowitz, ‘Note on the drawing power of crowds of different size.’, *Journal of personality and social psychology*, **13**(2), 79, (1969).

[12] James D Miller, ‘Effects of noise on people’, *The Journal of the Acoustical Society of America*, **56**(3), 729–764, (1974).

[13] Rafael Pax and Jorge J. Gómez-Sanz, ‘A greedy algorithm for reproducing crowds’, in *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection, 14th International Conference, PAAMS 2016, Sevilla, Spain, June 1-3, 2016, Special Sessions.*, pp. 287–296, (2016).

[14] Norbert A Streitz, Carsten Röcker, Thorsten Prante, Daniel Van Alphen, Richard Stenzel, and Carsten Magerkurth, ‘Designing smart artifacts for smart environments’, *Computer*, **38**(3), 41–49, (2005).

[15] Jinghui Zhong, Wentong Cai, Linbo Luo, and Haiyan Yin, ‘Learning behavior patterns from video: A data-driven framework for agent-based crowd modeling’, in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 801–809. International Foundation for Autonomous Agents and Multiagent Systems, (2015).

Virtual Development of a Presence Sensor Network Using 3D Simulations

Rafael Pax[1], Marlon Cardenas Bonett[1], Jorge J. Gómez Sanz[1], and Juan Pavón[1]

Universidad Complutense Madrid,
rpax,marlonca,jjgomez,jpavon@ucm.es,
http://grasia.fdi.ucm.es
28040 Madrid, Spain

Abstract. Testing the control and deployment of large networks of sensors and actuators is a complex and expensive task. This paper presents a 3D simulation tool that facilitates testing and measuring this kind of systems in a virtual environment, which alleviates the costs of doing these tasks in a physical setting. This is illustrated with an example of how a presence detection system can be designed to monitor the behavior of a crowd under different stimulus. The simulation does not only involve the devices, but provide input data so that they can be decoupled and analyzed separately. This decoupling allows to experiment with different deployments of sensors while the simulation is still working and evaluate their performance in real time. Such feature can be of assistance for decision making when designing a large installation or improving one. The paper contributes with a proof of concept where a large installation, together with its inhabitants, is simulated. The simulation is used then to create different simulations of photoelectric devices that register the proximity of an individual. The results permit to evaluate networks of such devices and think of different configurations without the limitations of the physical environment and the privacy and integrity concerns of individuals.

Keywords: Smart cities, simulation, sensors, actuators

1 Introduction

There is little support for smart environments design. A relevant issue is the testing of the control and deployment of large networks of sensors and actuators. This is a complex and expensive task when done directly with physical devices and involving people (e.g., actors). There are even some cases where the experimentation is risky (e.g., fire, falls, etc.) This paper presents a 3D simulation tool that facilitates testing and measuring this kind of systems in a virtual environment, which alleviates the costs of doing these tasks in a physical setting.

This paper introduces an architecture for the design of large installations that can be the first step to a large scale simulation of smart cities. The architecture allows to run multiple simulations and evaluate the performance of smart devices connected to it. It is assumed that simulated and real devices offer an API towards other components. In the simulation, simulated devices implement an API in different ways so as to capture what makes each device unique. As a proof of concept, the paper focuses on the creation of a network of presence sensors that satisfies the demands of a client that wants to exhaustively measure the presence in an area.

The case study introduces a simulation that corresponds to different occupations of an area. The simulation has a population of individuals that move through the installation following their daily activities. The client wants to evaluate whether a cheap network of sensors can achieve a quality presence tracking in the installation. The sample sensor network is evaluated and its precision measured.

The organization of the rest of the paper is as follows. Section 2 introduces the architectural solution that allows to run multiple devices. Section3 is a proof of concept where different simulations of the equivalent of a photoelectric sensor are designed. The related work section 4 shows the connection between this approach and other works. The paper finishes with a conclusions in section 5.

2 Simulation architecture

The simulation consists of different types of nodes, which are organized in four *tiers*, as shown in figure 1: application, sensor, simulation, and physical. All tiers can be executed on different computation nodes, which facilitates the scalability of the model. This can be easily extended as new application and sensor components can be deployed in separate nodes.

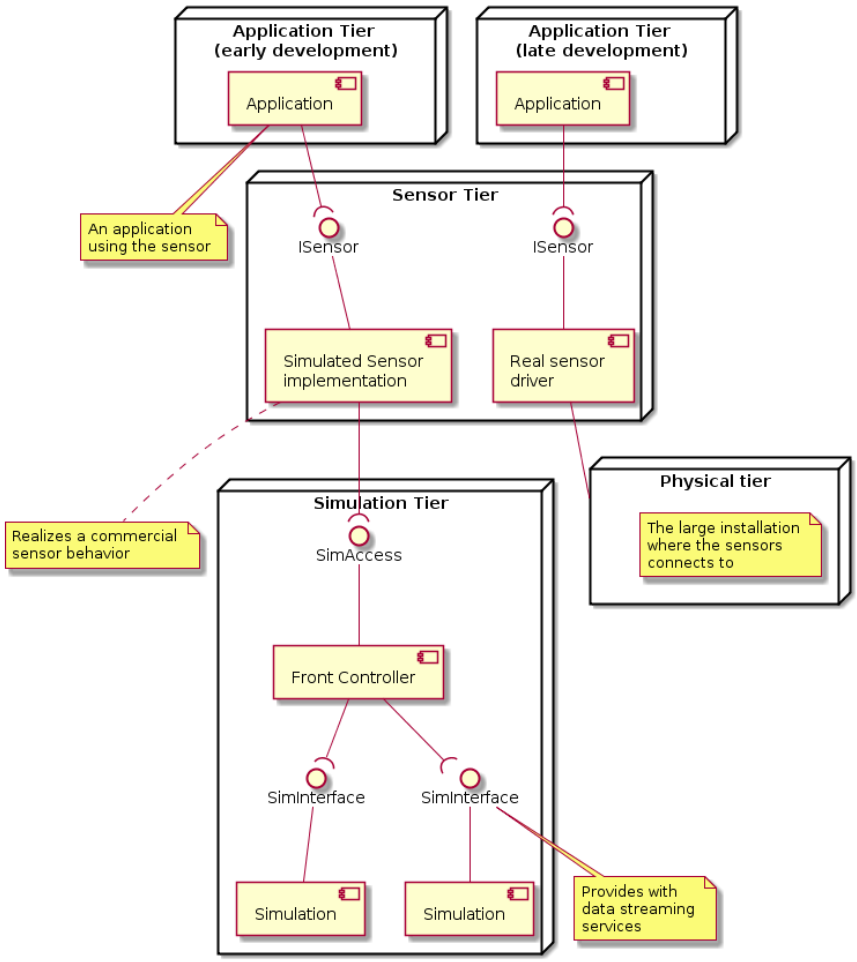

Fig. 1. The different tiers involved in the large installation sensor network development

This architecture induces a two phases development process, early development and late development. During early development, the work is done against a simulation of the large facility. The simulation provides the input for simulated sensors. These simulated sensors are expected to capture the behavior of real (commercial) ones, faulty behaviors included if needed. Late development assumes the application logic is already or almost built. Then, it can be connected to real sensors and proceed to final adjustments that take into account particularities of the sensors.

For this development cycle to work, the fidelity of the simulated sensor components has to be close enough to real ones, though maybe not exactly the same. Hence, simulated sensors provide the same, or similar, functionality, while real sensors connect to the real world.

Under this assumption, applications can work with both real or simulated sensors. It makes the development easier (and with lower costs) to work initially with simulated sensors and later on evolve to work with real sensors. Once the mission of the application is understood and what sensor layout is required, real deployments can take place with final tuning of the solution to specific contexts.

The tiers architecture reflect this development philosophy through the distinction of three levels of abstraction. This section will explain in detail the sensor, application, and simulation tiers. The physical tier is expected to be the infrastructure that is available for the late development stages.

2.1 Simulation tier

Previous work [GSCPC16] has shown that using data from simulations can be very useful for testing purposes before a deployment of devices. However, the computational cost of crowd simulations is tighly related to the number of agents (virtual avatars representing people and their behaviors) present in the simulation. Also, the realism of the simulation itself, such as 3D processing, physics or animations increases this cost.

Using this approach might not be affordable for everyone. A good approach for solving this issue is to execute the simulations in a central server, and allowing remote access to the simulation data from lightweight clients, via well-known data-transmission protocols and standards.

The simulation management is done through a front controller that mediates all accesses to the different running simulations. Figure 1 depicts two simulations as an example. The goal is to run in the server as many simulations as required, each one representing particular configurations of the facility. Current version runs the simulations in the same server, but it does not have to be that way. Simulations can be distributed across several computation nodes, which facilitates increasing the number of supported simulations.

A configuration of the simulation includes a 3D description of the physical environment, a population of simulated individuals, and behaviors associated to those individuals. These simulations run in headless mode (i.e. without a GUI) to make a better use of the machine resources. Removing rendering greatly reduces the workload and does not affect the results, since collision detection is still in place despite the absence of rendering.

External clients, in this case, will be the simulated sensors. They will obtain simulation data through a publisher-subscriber way. The client specifies the type of events that need to be sent over the network and then it will just listen to the data stream. For instance, if the sensor being simulated is a people counting sensor placed inside a carpet, 2D data is sufficient. But if the sensor being simulated is a more advanced sensor, such as a motion sensor, the information gathered from the simulation should be more accurate, and 3D information should be provided.

The information supplied can include the physical elements location in the simulation, as well as mobile elements. The information that is currently available from the simulator interface is the position of an element, its orientation, its velocity, mass, current animation (in the case of a person), and the type of element that it is (wall, door, person). The floor map information also provides information about building levels, if any. This selective accuracy and variety of information is relevant to support a higher number of sensor devices.

All information originated in the simulation has a time-stamp. Each event comes with a time-stamp that permits to measure the asynchrony between the data received by the different sensors. Required data bandwidth depends upon the number of elements as well as the required detail level. For the case study in section 3 the required bandwidth per sensor ranges from 3.4kB/s to 3.7kB/s. The simulation time can be shared across sensors so that they provide a consistent real-time-like vision to the application layer. However, there maybe still a gap between the time as perceived by the application and the time as represented in the simulation. This is something to be improved in new versions of the architecture.

To save bandwidth, a connection made from the client starts with an initial description of the environment. Afterwards, only changes to the initial state are provided. Changes are transmitted over the network labeled as events. Again, the reason is to reduce information overload.

2.2 Sensor tier

The sensor tier provides real and simulated sensors. Both offer the same interface towards the application, but feed on different data sources. On the one hand, software components representing real sensors are expected to use specific drivers that connect to the physical tier. On the other hand, simulated sensors capture essential features of the real ones and feed on the simulation data output to produce the expected outcome.

The particular implementation of the simulated sensor depends on the needs of the development. It has access to all the simulation events, and it has to process it in a way that the output matches the one expected from a real sensor. Some simulated sensors can include faulty behaviors or use different approaches to reproduce the real sensor behavior.

This decoupling from the simulation is important in order to separate responsibilities in the development. Simulations are already too complex and the inclusion of sensor devices within makes the problem even harder. Also, this allows to test at the same time different configurations or deployments of the same sensor type. Each configuration could be run at the same time and feed on the same data as the rest.

2.3 Application tier

The real goal of the development is to create the application that uses the sensor data. The development challenge consists in determining how many sensors, and of what kind, are needed to provide the service. To determine this, a fast method is needed that allows to quickly deploy and test sensor networks and evaluate if the provided information is enough to implement the target service. The cost of the sensor network is an important factor as well. There are sensors that provide the exact output that is required, but are too expensive to deploy a network of them. Finding the right combination requires time and experimentation.

The two phases development as proposed in this paper requires the application to assume that there are common interfaces that connect the application to whatever sensor, be it a simulated or a real one. In this way, the application will ignore whether it is connected to the real world or the simulated one.

The performance of the application depends also on the considered scenarios. The decoupling of simulations permits to have different simulations running at the same time, each one capturing crowd simulation scenarios of interest for the development within the simulated facility. Since simulated sensors are connected opportunistically to selected simulations, this allows the application to selectively connect to running simulations and evaluate its performance directly. Besides, this operation would be made in similar conditions as it would be in the real world, since the application does not know if it is connected to real world sensors or not.

Finally, the different tier decoupling allows to run different instances of the same application. This contributes to accelerate the development since all work is not centered in a single computer.

3 Case Study

This practical case deals with the construction of a network of sensors to detects the presence a set of individuals in a physical space, which is simulated by a computer. With this purpose, different presence sensors and configurations are studied.

The sensors can use ultrasonic, optical, inductive, magnetic, and capacitive technology. Those with the longest range are the optical and ultrasonic. Ideally, we want to consider sensors that are allocated only into a single location and do not require a reflector.

Ultrasonic are expensive, but have the longest range (up to 8 meters). Price of ultrasonic devices is high (starting from 300 euros a piece). They are not affected by ambient noise or light. Some optical sensors do not require either a reflector and work with an emitter and a receiver that captures reflected signals. A pulse is emitted and, if received, the presence sensor is activated. There is some tolerance to ambient light noise and flickers from fluorescents. The drawback is their short range, which is a maximum of 1 meter.

The pulse sensor can be modeled as an element that detects objects within a semi-sphere whose center is the sensor and the radius is one meter. The model can be improved, but serves enough for this paper and to evaluate the expected performance of a presence network sensor.

The simulation will start a random stream of people that will enter the facility, in this case one of the floors of the building of the Faculty of Computer Science where we work, which is shown in figure 2. This figure shows the 3D rendering of the simulation for the sake of clarity only, since, in the simulation architecture as shown in figure 1, these simulations usually run in headless mode.

Within the facility, six sensors are located in one of the more traveled corridors, which contains alternate paths or detours to classrooms that will change the movement of individuals.

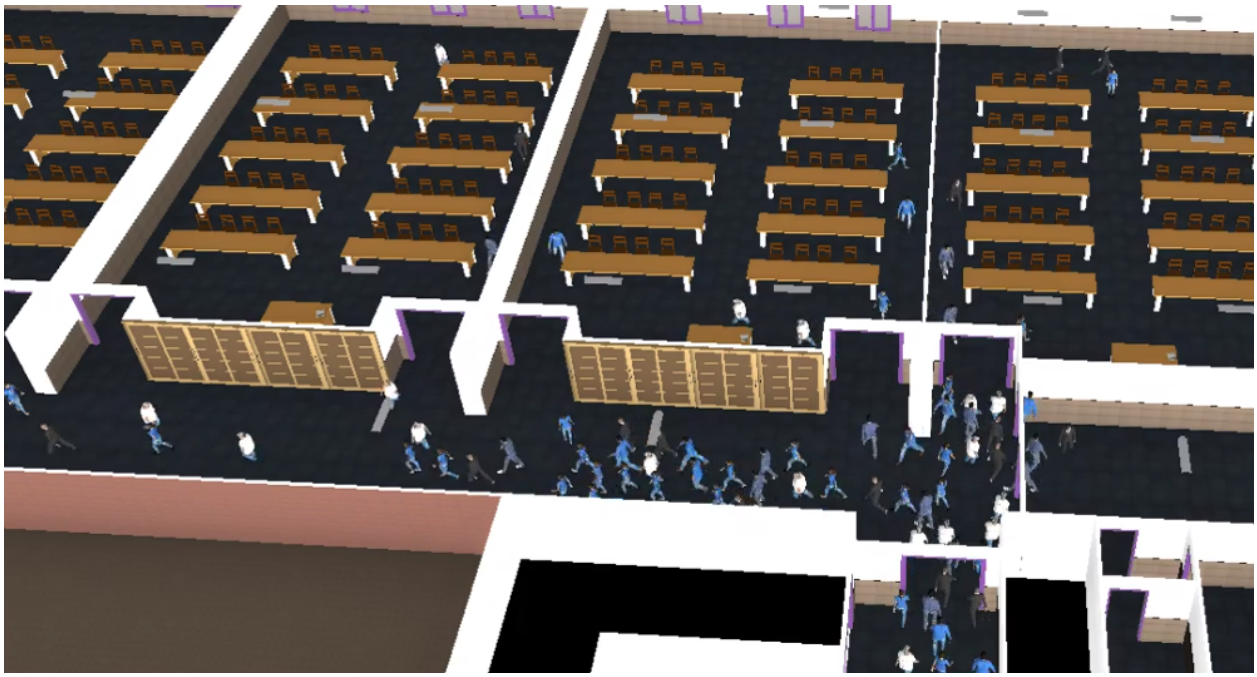

Fig. 2. 3D rendering of the simulation case study

The simulated sensors in the sensor tier within this development are provided with a GUI for drawing what information each simulated sensor is accessing. Figure 3 shows this 2D map, which is produced from the 3D simulation run as in figure 2. The 2D map shows the x and z axis coordinates of the location pointed out by the mouse. The upper right corner shows the current simulation time. Green circles represent hollows in the walls (windows or doors). Blue dots represent individuals traversing the installation. The red filled circles represent physical objects in the area. Sensors S_1 to S_6 are labeled in the figure at the center of the corridor. The location of the sensor is depicted as a red dot and the range of the sensor, one meter, is drawn as a concentric circle. The controlled area is drawn as an orange rectangle wrapping all the sensors.

The six sensors will be deployed across this orange rectangular region (see figure 3). Each sensor is allocated at the ceiling of the facility and has a range of one meter. The sensor is activated if an individual crosses the hemisphere whose center is the sensor location. The hemisphere that models the range of detection of these sensors will detect those pedestrians who are high enough to be within one meter of the ceiling, when walking underneath. To measure the simulated sensor measurement efficacy, a rectangle covering the corridor is drawn and the people walking across it measured.

The objective is to find the best configuration and distribution of the sensors to build a mesh with a specific number of devices that will allow to detect as accurately as possible the individuals that transit

within a certain space. Comparing the sensor detection against the orange rectangle based approach permits to evaluate the precision in the detection.

If the six sensors are numbered from S_0 to S_6, we can model the application that combines their output as equation 3. Hence, there will be people in the corredor if any sensor detects anyone.

$$app = S_0 \vee S_1 \vee S_2 \vee S_3 \vee S_4 \vee S_5 \vee S_6 \tag{1}$$

If the function that determines if there is really someone at the corridor is named h, the purpose of the experiment is to assess the suitability of a location and a number of sensors so that equation 3 is satisfied.

$$app \approx h \tag{2}$$

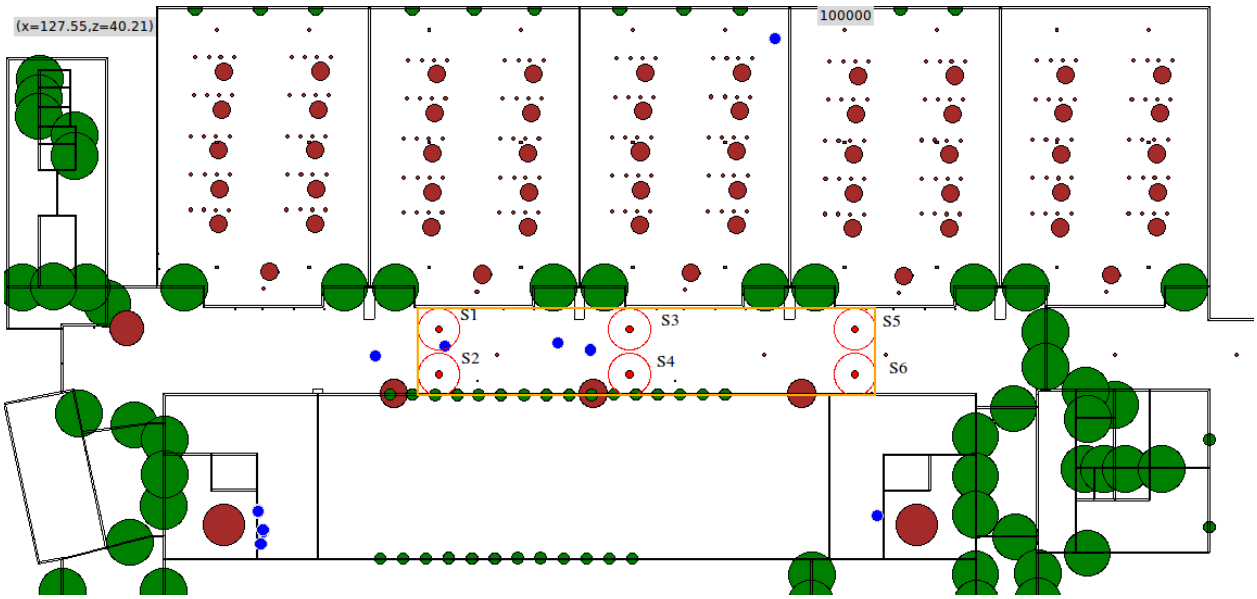

Fig. 3. Location of sensors in the case study

The figure 4 shows the account of people within the sensor deployment area, the orange rectangle in figure 3, but divided into 6 pieces, each one corresponding to the proximity of an individual sensor location. In blue, there is an account of the number of times a person crossed the area assigned to each sensor. In orange, the diagram depicts the number of persons crossing the area when the sensor activated. It can be seen that the actual number of people crossing the orange rectangle is greater than the number of people triggering the sensors.

There are three hypothesis that explain what is happening. Firstly, the control area is bigger than the range of the sensors. Hence, necessarily, there will be more people crossing the area than people being detected. Secondly, it turns out that some of the individuals did cross the section, but never got close to some sensors. The reason is that these persons crossed doors to enter other rooms and avoiding the sensor location. For instance, a person entering the room close to S_3 may never enter within the S_3 range, but will be accounted by the orange rectangle. Thirdly, characters in the simulation behave following the shortest path. Hence, if characters intend to enter into the room close to S_1, they can enter directly without triggering S_1 and S_2. The sensor S_6 surely is not fired frequently because of the same reason, because the shortest path crosses more times S_5 than S_6.

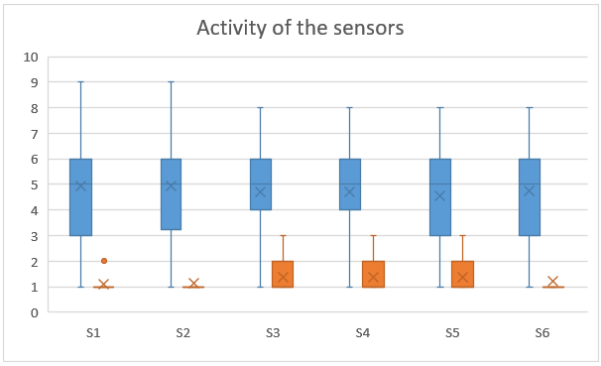

Fig. 4. People present in the proximity of the sensor (blue) vs people present when the simulated sensor triggered (orange). X axis represent each sensor. Y axis represents the number of persons.

Despite the low number of people crossing the sensor range area, the total outcome of the presence function is not unsatisfactory. Figure 5 contains the representation of the evolution of *app* and h functions. In general, h returns more positive activations than *app*. It has already been discussed why this could be, and the reason maybe an unexpected behavior of the pedestrians, that did not traverse the corridor as expected, but got into classrooms and managed to avoid sensors. Nevertheless, the figure does not shows false positives and fails when the pedestrians show this unexpected behavior.

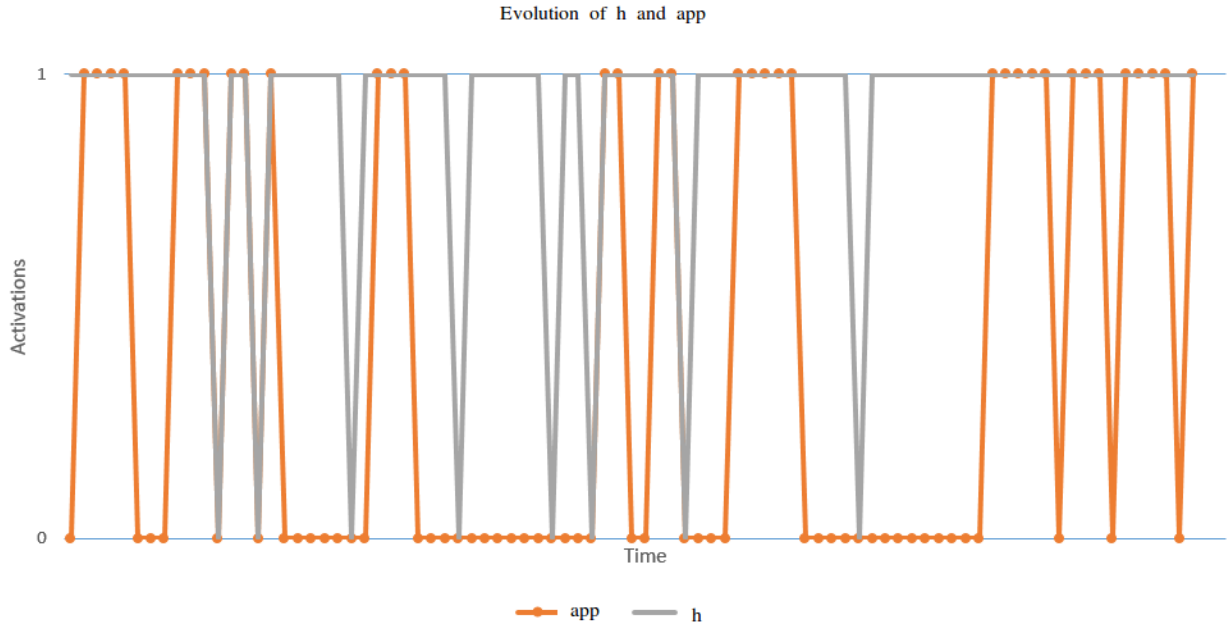

Fig. 5. Representation of the evolution of h against *app*

The lesson from this experiment is that the location of the sensors at the ceilings is not a bad idea, but that the behavior of the pedestrians greatly affects the success ration. We think it is possible a better location of the sensors so that better success ratio is obtained.

4 Related work

The work [DA09] studies the energy savings in a building based on occupancy sensors. They use acoustic, light, motion, CO_2, temperature and relative humidity sensors. This paper has been proposed with a motion sensor in mind only. It limits then to one kind of sensor and discovering patterns in the data was not the goal. The paper [DA09] also uses a simulator, energyplus, to test the outcome of the occupancy detection with respect the energy consumption in the building. In this work, the simulation is used to generate data for the sensors, and no actuator is defined.

The paper [PGZ+13] introduces the Tokyo Virtual Living Lab for conducting controlled driving and travel studies. The goal is to raise awareness in order to create an eco-sustainable and optimized transportation. The participation of users in the simulation allows to foresee the effect of traffic and experiment with different ways of driving styles. This paper does something similar, since it intends to use a 3D simulation as testbed for testing ideas. However, this paper focuses on devices and does not involve human participation.

The work [HHJ+16] point at 3D as a useful tool to support urban operational decision makers. The work uses 3D models to represent the outcome of some natural, such as floodings, or artificial events, energy outages. The system feeds from real data to provide a mixed reality perspective. Our approach is different in that the simulation is not fed with external data, but it creates the different data streams through the aim of game engines. Also, it facilitates the development of devices. Simulated devices are connected to the simulation and they can be used to service other applications. Then, the applications, if the interface to the simulated sensors is sufficiently accurate, can switch the kind of sensors used and connect to real ones, if available. In this way, the same application can be reused in a different context.

The work [CDSB14] uses Ubiksim to simulate a Smart Campus and study how an agent based social simulation model can help and influence decision making processes in the context of the faculties management. The differences with the current proposal is that Ubiksim is based on 2D engines that do not account for three dimensions. Hence, situating a device in the ceiling or at some height is not explicitly accounted. Also, the height of the characters or peculiarities in their body movement are not captured, so this particular problem where sensors detect presence within the detection area of a sensor (represented by a cone) would not have been possible.

[RMDK11] introduces a tool chain where the environment is reproduced through a 3D simulation created with SweetHome3D software, as in Ubiksim. Then, they use a middle-ware based on ROS (Robot Operating System) that decouples the application from the simulation and permits to switch between real sensors and simulated sensors. The physical reactions in the simulation are achieved through a plugin called Gazebo. The authors have to determine some consequences due to the use of limited physics. This approach is similar to ours. In fact, early work was made with SweetHome3D [PP16] and cannot achieve the performance required for thousands of characters. Besides, SweetHome3D does not include physics either, and that forced authors to use Gazebo. We have solved this by using a game engine, JMonkey, and a physics engine, Bullet. These elements may suggest that this approach is more suitable towards the simulation of large spaces.

5 Conclusions

The paper has introduced an architecture that enables the simulation of the equivalent of optical presence sensors in a large installation. The network is used to measure the presence in a large installation through cheap sensors that are opportunistically distributed. This allows to choose which areas and with what precision presence has to be detected beforehand. The paper has shown how such network can be simulated against different crowd simulation configurations. Also, how the performance of the simulated devices and their efficacy can be measured to imply how well the actual deployment works. It has shown as well the importance of including crowd simulation elements that permit to test the idea in working conditions closer to real ones. In particular, it served to identify issues with an intuitive six sensor deployment across a corridor.

The paper has not addressed which sensor behavior is more suited towards this task. It is assumed that the developer will reproduce the behavior of different commercial sensors and translate them to the simulation. Then, a suitable combination of them can be used to solve the problem or, at least, to know more. The size

of the crowd simulation is a modest one of some hundreds of individuals. More experiments have been made with thousands, but it is still pending work to achieve magnitudes of populations like those to be found in cities.

6 Acknowledgements

We acknowledge support from the project "Collaborative Ambient Assisted Living Design (ColoSAAL)" (TIN2014-57028-R) funded by Spanish Ministry for Economy and Competitiveness; and MOSI-AGIL-CM (S2013/ICE-3019) co-funded by Madrid Government, EU Structural Funds FSE, and FEDER.

References

[CDSB14] Francisco Campuzano, Ioannis Doumanis, Serengul Smith, and Juan A Botia. Intelligent environments simulations, towards a smart campus. In *2nd International Workshop on Smart University*, 2014.

[DA09] Bing Dong and Burton Andrews. Sensor-based occupancy behavioral pattern recognition for energy and comfort management in intelligent buildings. In *Proceedings of building simulation*, pages 1444–1451, 2009.

[GSCPC16] Jorge J Gómez-Sanz, Marlon Cardenas, Rafael Pax, and Pablo Campillo. Building prototypes through 3d simulations. In *Advances in Practical Applications of Scalable Multi-agent Systems. The PAAMS Collection: 14th International Conference, PAAMS 2016, Sevilla, Spain, June 1-3, 2016, Proceedings*, volume 9662, page 299. Springer, 2016.

[HHJ+16] Shaun Howell, Jean-Laurent Hippolyte, Bejay Jayan, Jonathan Reynolds, and Yacine Rezgui. Web-based 3d urban decision support through intelligent and interoperable services. In *Smart Cities Conference (ISC2), 2016 IEEE International*, pages 1–4. IEEE, 2016.

[PGZ+13] Helmut Prendinger, Kugamoorthy Gajananan, Ahmed Bayoumy Zaki, Ahmed Fares, Reinaert Molenaar, Daniel Urbano, Hans van Lint, and Walid Gomaa. Tokyo virtual living lab: Designing smart cities based on the 3d internet. *IEEE Internet Computing*, 17(6):30–38, 2013.

[PP16] Rafael Pax and Juan Pavón. Agent-based simulation of crowds in indoor scenarios. In *Intelligent Distributed Computing IX*, pages 121–130. Springer, 2016.

[RMDK11] Luis Roalter, Andreas Moller, Stefan Diewald, and Matthias Kranz. Developing intelligent environments: A development tool chain for creation, testing and simulation of smart and intelligent environments. In *Intelligent Environments (IE), 2011 7th International Conference on*, pages 214–221. IEEE, 2011.